Universidade de Aveiro
2011

Departamento de Electrónica, Telecomunicações e Informática

Filipe Miguel Mendo Duarte

Interacção com ecrãs públicos através de um dispositivo móvel

Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática
2011

Filipe Miguel Mendo Duarte

Interacção com ecrãs públicos através de um dispositivo móvel

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Professor Doutor Paulo Miguel de Jesus Dias, Professor Auxiliar, e da Professora Doutora Maria Beatriz Alves de Sousa Santos, Professora Associada com Agregação do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

o júri

presidente

Doutor Joaquim João Estrela Ribeiro Silvestre Madeira
Professor Auxiliar do Departamento de Electrónica e Telecomunicações e Informática da Universidade de Aveiro

vogais

Doutor Frutuoso Gomes Mendes da Silva
Professor Auxiliar do Departamento de Informática da Escola de Engenharia da Universidade da Beira Interior

Doutora Maria Beatriz Alves de Sousa Santos
Professora Associada com Agregação do Departamento de Electrónica e Telecomunicações e Informática da Universidade de Aveiro

Doutor Paulo Miguel de Jesus Dias
Professor Auxiliar do Departamento de Electrónica e Telecomunicações e Informática da Universidade de Aveiro

palavras-chave

Interacção Humano-Computador, Ecrãs públicos, Ecrãs de grandes dimensões, Usabilidade, Computação móvel

resumo

O uso de ecrãs de grandes dimensões para a exibição de conteúdo informativo contém limitações a nível de interacção que devem ser superados. Os dispositivos móveis actuais, equipados com acelerómetros, bússolas digitais, e ecrãs sensíveis ao toque, constituem uma alternativa para permitir esta interacção sem a necessidade de hardware adicional, uma vez que os utilizadores transportam o periférico consigo.

Esta dissertação apresenta os passos necessários ao desenvolvimento do DETI Interact: um sistema informático para exibição de conteúdo informativo para o Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro, que permite a utilizadores usarem um dispositivo móvel Android para interagir com os conteúdos apresentados. Após o desenvolvimento foram realizados testes com utilizadores e o resultado final foi colocado em funcionamento nos ecrãs de grandes dimensões presentes na entrada do departamento.

keywords

Human-Computer Interaction, Public Screens, Large Screen Displays, Mobile Computing, Usability

abstract

The use of large screen displays for the presentation of informative content has interactivity limitations that should be overcome. Current mobile devices come equipped with accelerometers, digital compasses, and touchscreens that constitute an alternative to allow interaction with these displays without the need for additional hardware, since the users will be carrying the peripheral with them.

This dissertation presents the necessary steps in the development of DETI Interact: an information system for the exhibition of informative content from the Department of Electronics, Telecommunications, and Informatics of the University of Aveiro, which enables users to use an Android mobile device to interact with the displayed contents. After its development, it was submitted to user testing and it is now operating in the large screens present in the department.

Table of Contents

Index of Figures

Acronym List

API Application Programming Interface

ASP Active Server Pages

CLR Common Language Runtime

CPU Central Processing Unit

DETI Departamento de Electrónica, Telecomunicações e Informática

DSD Distribuição de Serviço Docente

FCL Framework Class Library

GDI Graphics Device Interface

GPS Global Positioning System

GPU Graphical Processing Unit

HTTP Hypertext Transfer Protocol

IDE Integrated Development Environment

IIS Internet Information Services

LCD Liquid Cristal Display

LINQ Language Integrated Query

MFC Microsoft Foundation Class

MVVM Model-View-View Model

NFC Near-Field Communication

NUI Natural User Interface

OEM Original Equipment Manufacturer

OS Operating System

SDK Software Development Kit

UI User Interface

URI Uniform Resource Identifier

URL Uniform Resource Locator

UX User Experience

W3C World Wide Web Consortium

WCF Windows Communication Foundation

WPF Windows Presentation Foundation

WSDL Web Services Description Language

1 Introduction

1.1 Motivation

As technology evolves, computer and television screens have become increasingly thinner and cheaper. In addition, some of these screens are even equipped with touch-sensitive panels enabling interaction via touch input. This has allowed these screens to be used in public locations such as airports, train stations, malls, museums, lobbies of buildings, etc., presenting relevant information to the people passing by. The use of these screens, however, presents a challenge in how information is displayed and how people can interact with it. When the screens are touch-sensitive it is relatively easy to make the system interactive, however, when they are not, the system usually displays a predetermined set of information, usually presented in a looping or alternating sequence. The objective behind DETI Interact is to study how to make these screens interactive, allowing the user to control the flow of information and choose what he wishes to see.

Interaction with these screens can be achieved in a variety of ways, as can be seen in several projects presented in Chapter 2. These projects research the use alternative input peripherals, such as a smartphone, to more advanced scenarios such as video tracking or motion capture. DETI Interact is a follow-up project to DETI Guide (Palha 2010), a previous study that investigated how a smartphone could be used to interact with a public screen, for this reason, the chosen approach was to use a smartphone. The result is that there will be no hardware changes required on the system (for instance, no tracking cameras, touch panels, etc.), since users will be carrying the interaction peripheral themselves. However, the choice of how interaction is done is only one of the steps.

Given the nature of the device, several aspects concerning the interaction have to be addressed with care: the way the content is displayed on the screen, the way the user interface is designed, and the way the device interacts with the content, are all important factors, which will be addressed in this dissertation.

1.2 Objectives

DetiGuide(Palha 2010) was a system developed in the scope of a previous dissertation, which studied the possibility of using of a smartphone to interact with the screens in the lobby of the Department of Electronics, Telecommunications, and Informatics of the University of Aveiro. The present dissertation intends to go further. Following the results from the previous study, this project aims to develop of a platform for displaying interactive information on those screens, and integrate the possibility of interacting with this information via a smartphone.

This project has two main objectives:

- System architecture: the existing DSD (*Distribuição de Serviço Docente*) Platform which supplies information to the system that is currently operating in the lobby of the department should be studied in order to use the supplied information to develop interactive contents.

- Interaction with the displays: the developed system should allow a user in the lobby to interact with the screen to access relevant information, such class times, information regarding the teachers, etc. In addition to planning the system for communication with mobile devices, the system should also be prepared to communicate with other devices, such as a WiiMote or cameras via body tracking. It should also be prepared to run on computers with touch-sensitive screens.

The resulting developed system will be deployed to the computers in the lobby of the department where users will be able to interact with the system using their phones.

1.3 Dissertation Structure

This dissertation consists of five chapters, excluding this introductory section.

In chapter 2, “State of the Art”, the bibliographic research done for the development of this system is presented. It encompasses several projects that propose different methods of interaction between users and large screens.

Chapter 3, “System Design”, illustrates the different steps in the preparation of the development of the system. It describes the used tools, studies features that were chosen to integrate the system, as well as some of the required APIs, and presents the architecture of the system, leading to the next chapter.

In Chapter 4, “Development”, the development phase of the project is described. Some implementation details as well as problems are explained in order to illustrate how the core of the system works, and how it can be modified or extended.

Chapter 5, “User Evaluation”, explains how the developed system was tested. The results of the experiments are presented, and are discussed as to how they are used to improve the system.

Finally, in Chapter 6, “Discussion”, the work is reviewed, along with the results gathered from the testing sessions. Some possibilities for future work on the developed system are discussed.

2 State of the Art

2.1 Introduction

With advances in technology making computer screens increasingly thinner and cheaper, these became more common in public places such as shops, cafes, museums, atriums of buildings, and train and subway stations. However, these screens are often used to display information, rather than allowing users to interact with it. These public displays fostered several research projects to investigate interaction between users and the displayed information using mobile devices, augmented reality, and motion capture.

The projects studied for the development of DETI Interact propose diverse ways for interaction with information presented on computer screens. Apart from the many differences between these projects, some recurring elements allowed these studies to be grouped into three distinct categories. The first group of projects focuses the use of smartphones for interaction, concentrating on the technologies that can be used for communication between devices such as Wi-Fi, Bluetooth, and SMS. They present different methods of communicating data between mobile devices and large screens in order to interact with some part of the system. The second group of projects uses different approaches of video tracking and augmented reality to allow interaction between mobile devices and the target screens. Finally, the third group uses motion capture for the interaction, eventually dismissing the use of mobile devices altogether.

2.2 Communication between devices

Since DETI Interact will involve communication between a mobile device and a computer, the several means of achieving this communication must be studied, allowing for a more educated decision.

Gmote (Stogaitis and Sun 2008) is an example of an Android application that enables interaction between a mobile device and a computer, using a Wi-Fi connection. For the devices to communicate an application that acts as a server is deployed to the computer and a client application is executed on the mobile device. This system supports most of the media playback related features found in a standard remote control such as play, pause, rewind, volume control, etc. as well as new features like file browsing and media streaming. It can also use the mobile device's touchscreen as a touchpad, as well as Android's software keyboard as a keyboard, enabling the user to control the mouse pointer and keyboard remotely. Similar applications such as RemoteDroid, BoxeeRemote, TivoRemote, and SqueezeControl, have also been developed that allow interaction between additional mobile and desktop operating systems (Wired 2009).

While Gmote can be used to interact with a computer, similar applications have been developed that allow interaction with a television or a set-top box. Samsung has developed the Internet@TV platform (Samsung 2010), a service that allows users to access online content such as photos, video, music, social networks, news, games, etc., on an internet-enabled Samsung TV or Blu-ray player. Interaction with the system can be done with a regular remote control, however, Samsung has developed an application for iOS and Android-based devices, which lets users interact with the television or set-top box using their mobile phone. Google has also developed a similar system, Google TV (Google 2010), which also supports the use of a phone to control the device. In addition to typical functions of a remote control, Google's phone application also enables streaming video and audio content to the television.

Other systems, such as Touch & Interact (Hardy and Rukzio 2008; Hardy and Rukzio 2008), allow users to interact with large displays using mobile devices with NFC (Near Field Communication) technology. NFC consists of a set of short-range wireless technologies, such as RFID, typically relying on radio frequency for communication. The display used in this system is attached to a mesh of NFC tags, which enables the phone to detect the tags when close to the display. Given the nature of NFC tags, the system supports gestures such as hovering the phone over a certain tag, or touching an area close to the tag. The phone is used to identify the tags on the display, and communica-

tion with the system is done using Bluetooth. Figure 2.1 shows a user selecting cells on a map application on a display using Touch & Interact.

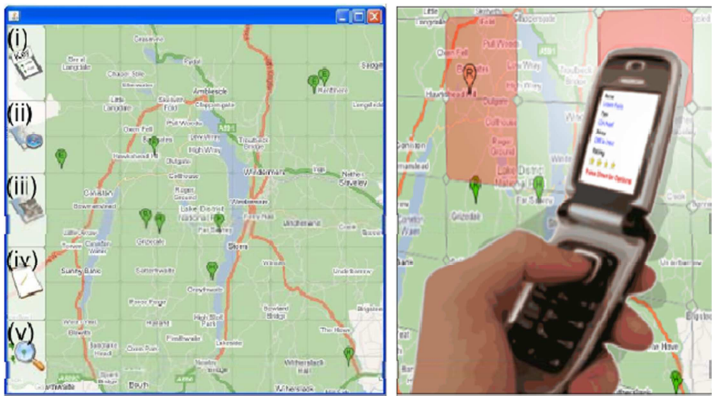

Figure 2.1 - A tourist guide application using Touch & Interact (Hardy and Rukzio 2008).

The principle behind Touch & Interact also produced two additional projects: Touch & Connect and Touch & Select (Seewoonauth, Rukzio et al. 2009). Touch & Connect allows users to transfer files between a laptop and an NFC phone simply by touching the NFC tag placed on the laptop. The system then configures the connection and transfers the date, without any additional user input required. For Touch & Select, a mesh of NFC tags similar to the one used in Touch & Interact allows the user to select items on the laptop by touching its display with the mobile device. Communication with the system in either case is done via Bluetooth.

MAGICBoard (Tang, Finke et al. 2008) is a public digital forum designed to encourage bystanders to interact with the system. Bystanders are individuals that are close to the large display but never actively participate in using the system. The system allows users to view or discuss on popular topics, and participate in some voting sessions. Interaction with the system was achieved by using either SMS messages, or a nearby kiosk terminal. An example of a deployed MAGICBoard application can be seen in Figure 2.2. Polar Defence (Finke, Tang et al. 2008) is a tower defence game that was deployed in a similar setting, using the same system behind MAGICBoard. Users would play the game by sending SMS messages to the system with the appropriate instructions. These projects attempted to reduce barriers to interaction, such as embarrassment, trust and security, and encourage spectators to become actors in the system (Kaviani, Finke et al. 2009).

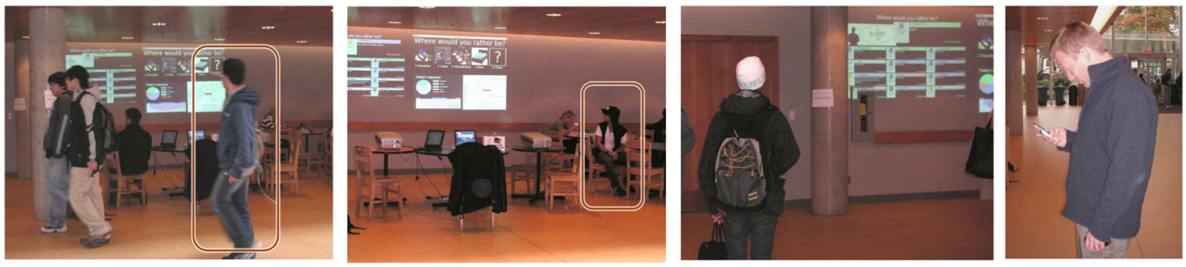

Figure 2.2 – A deployed MAGICBoard (Tang, Finke et al. 2008) system illustrating, from left to right, a *passer-by*, a *stander-by*, and *engaged bystander*, and a *contributor*.

As mentioned in Chapter 1, DetiGuide (Palha 2010), a project that preceded DETI Interact, studied interaction with large displays using a mobile device running Android OS. The system was deployed to computers located at the lobby of the Department of Electronics, Telecommunications, and Informatics of the University of Aveiro, and consisted of a web application similar to the department's website, and a mobile application, used to interact with the web application. The web application displayed information regarding how to navigate in the system through a series of widgets arranged along the edges of the web page. This information would assist the user in the manipulation of the mobile device in order to navigate on the web site. The mobile application supported several interaction techniques, such as using the accelerometer, gestures drawn on the touchscreen, and methods to control the mouse pointer. Communication between the mobile device and the web application was done over a Wi-Fi connection.

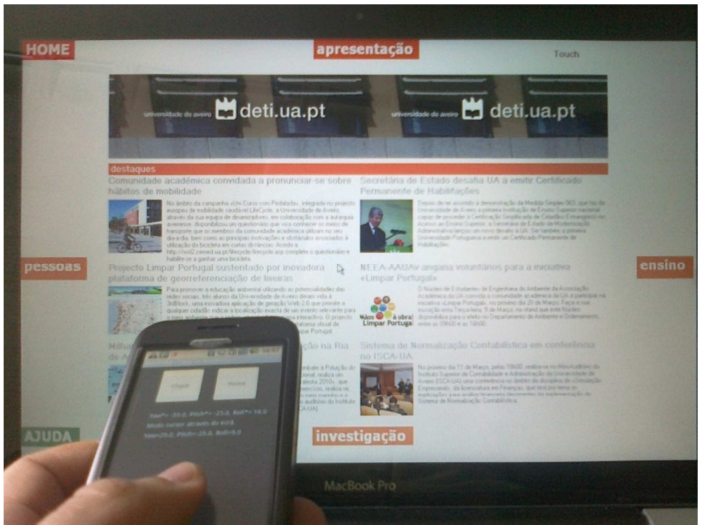

Figure 2.3 – User interacting with DetiGuide using the accelerometer on a mobile device.

In conclusion, the most used communication methods include Bluetooth, Wi-Fi, or SMS. For DETI Interact, Bluetooth and Wi-Fi present the most versatile alternative since there is no interference from the operator in the communication process.

2.3 Video Tracking

Interaction with large displays can also be achieved by tracking objects on a video feed, such as the movement of light sources. LasIRPoint (Cheng and Pulo 2003) uses an infrared tracking device and an infrared laser pointer in a presentation environment. The tracking device is coupled with the projector, facing the screen, so that it captures the gestures performed by the user with the laser pointer. Users can perform gestures such as selecting an object by drawing a circle around it with the laser pointer, and the system processes these gestures and proceeds accordingly.

C-Blink (Miyaoku, Higashino et al. 2004) uses a camera to track the movement of a mobile phone with a coloured screen (Figure 2.5). The phone runs an application that rapidly changes the hue of the LCD screen; this hue sequence is unique and easy to detect, and it can be used to identify multiple mobile phones, which allows the system to support multiple users. The camera is positioned above the display, and the user waves the phone in front of it (Figure 2.4). It uses the signal coming from the phone to track its position and uses this to control the position of the cursor on the display.

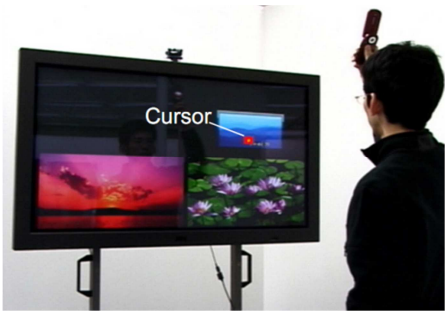

Figure 2.4 – Using C-Blink to control a display (Miyaoku, Higashino et al. 2004).

Figure 2.5 – C-Blink with an overlay of the video captured by the sensor (Miyaoku, Higashino et al. 2004).

In addition to identifying the user, a C-Blink signal can also encode extra information. To illustrate this, the system also implements several basic functions such as Click, Grab, and Pitch. Each of these functions is sent to the system through a C-Blink signal. For instance, if a user wants to download an image from the system (using the Grab function) he aims the device at the image and presses a button; this generates a C-Blink signal that encodes the terminal ID of the cell phone and along with a flag for the grabbing function. The system then responds by sending the image to the phone via the wireless network.

Touch Projector (Boring, Baur et al. 2009) is a system that uses an augmented reality solution allowing users to remotely manipulate content presented on remote displays on a mobile device via a live video image captured by the device's camera. The user aims the mobile device at a display and interacts with the image on the screen via touching and dragging the objects. Input to the touchscreen is "projected" to the selected display as if it had occurred there. The user can select an object and move it with a finger (Figure 2.6), which results in a relatively precise movement, or he/she can select and hold an object and move the device instead, which performs a more coarse movement. Objects can also be moved between displays by touching the object in the first display and moving the device towards the second, dragging the object off-screen and into the intended display.

Figure 2.6 – Selecting an object using Touch Projector (Boring, Baur et al. 2009).

In order to achieve a better user experience, features such as zooming and freezing the video image are also available. Zooming allows the user to interact with distant displays; this can be done automatically when the user aims at a display where the image does not fill the whole area of the video. Freezing the image allows for higher precision since the user will be dealing with a still image; it also eliminates the need to keep the device still or pointed at the screen, which avoids fatigue.

Madhavapeddy *et al.* (Madhavapeddy, Scott et al. 2004) also present a camera-based interaction using mobile phones, involving manipulating tagged interactive UI elements such as sliders and dials (Figure 2.7). Interacting with the system relies on using *SpotCodes* (Figure 2.8), a visual tag similar to a QR-Code, which is detectable by the camera-phones. These tags can be active (e.g. generated dynamically by the system) or passive (e.g. printed on paper). By placing the phone close

to the tag, ensuring the camera on the phone can identify it, the user can position a graphical slider, or orient a graphical dial by manipulation the camera-phone.

The mobile phone is responsible for tracking, and identifying the tags, while the system is responsible for displaying and refreshing the tracked images. Communication between the devices is achieved using Bluetooth.

Figure 2.7 – Madhavapeddy's *et al.* (Madhavapeddy, Scott et al. 2004): User interacting with a world map application.

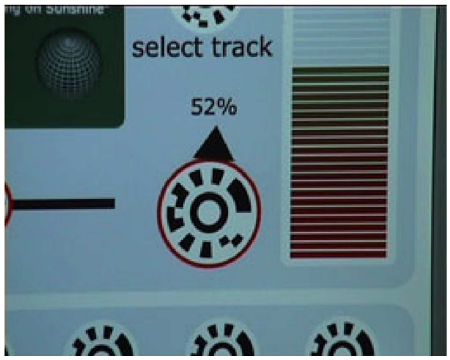

Figure 2.8 - Example of a dial tagged with a *SpotCode* using Madhavapeddy's *et al.* (Madhavapeddy, Scott et al. 2004) system.

2.4 Motion Capture

Other ways used to interact with information displayed on large screens include motion capture techniques. Motion Capture systems typically require specific cameras, such as Infrared or Stereoscopic cameras, as well as markers on the tracked objects to assist in the motion capture process.

Vogel *et al.* (Vogel and Balakrishnan 2004) developed a prototype system for use in interactive public ambient displays which allows for multiple users. Interaction with the system is performed using simple hand gestures (Figure 2.9) as well as touch screen input. A motion capture system is responsible for the detection of users and gestures, and requires the use of small passive markers on the body parts that require tracking. While this is inconvenient, advances in computer vision techniques should obviate the need for markers.

Figure 2.9 – Users interacting with the prototype system developed by Vogel *et al.* (Vogel and Balakrishnan 2004).

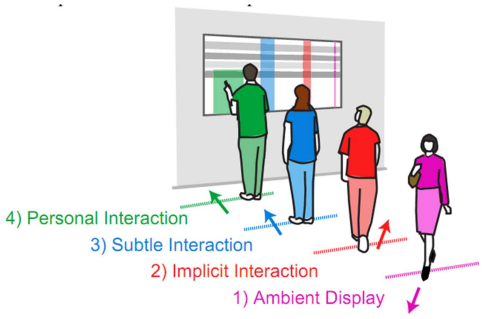

Figure 2.10 – Illustrating the four interaction zones covered by the system(Vogel and Balakrishnan 2004).

This prototype explores the concept of transitioning from implicit to explicit interaction with both public and private information. The system covers four interaction zones: Personal Interaction, Subtle Interaction, Implicit Interaction, and Ambient Display, as shown in Figure 2.10. When a user is far from the screen, the system will perform as an ambient information display, showing a range of categorized information. If the system detects that the user is closer and facing the screen, the system enters the Implicit Interaction phase, adapting the UI by displaying brief information concerning the user, as well as subtle notifications regarding personal or public information items that may require attention. As the user approaches the screen, the system enters the Subtle Interaction phase; the system adds more detail to the information and notifications onscreen. When the user selects an information item the system enters the Personal Interaction phase, the user is now in front of the screen, which allows for direct manipulation of the information.

Underkoffler (Underkoffler 2010) developed a system in which users would wear a set of gloves with reflective material placed on the tips of the fingers. The system would then be able to track the movement of each finger, which would allow the user to interact with large displays using hand gestures. This system was developed to be used on the sci-fi movie Minority Report, and has been updated to allow interaction with three-dimensional environments.

Orange, the UK based mobile network operator, unveiled a gesture based interaction screen, developed by The Alternative (TheAlternative): the Orange Interactive Window (GIZMODO) (Figure 2.11). The system uses cutting-edge motion capture technology that tracks gestures executed by the users to control the content on the screen without the need to place passive markers on the user (Figure 2.12). Other competitors in the mobile market such as Vodafone, Samsung, and Sony, have also developed their own versions of this interactive window.

Figure 2.11 – The Orange Interactive Window (GIZMODO): Main display.

Figure 2.12 – User interacting with the Orange Interactive Window (GIZMODO).

Kinect (Microsoft 2010), developed by Microsoft, also offers state-of-the-art motion capture technology. It is intended for home use on the Xbox 360 video game platform and eventually computers equipped with Microsoft Windows (Anthony 2010).

Figure 2.13 – The Kinect sensor device (Microsoft 2010).

The device, shown in Figure 2.13, includes an RGB camera, a depth sensor and a microphone array (Totilo 2010). The RGB camera is used as a regular video camera, and can be used for facial recognition. The depth sensor consists of an infrared laser emitter and a CMOS Active-Pixel Sensor. The emitter projects a mesh of infrared points to the surrounding environment, which the sensor captures; the distance between these points is used to tell how far a certain object is. By mapping the depth image with the image from the RGB camera, the system can create a 3D video image of the surroundings. This allows the system to provide 3D full-body motion capture under any ambient light conditions. The microphone array can suppress ambient sounds, and locate sound sources, which enables innovative voice recognition capabilities.

Figure 2.14 – Using Kinect with an application in Windows 7.

Figure 2.15 – Two users interacting with Kinect simultaneously.

The hardware behind Kinect along with its software technology enables the system to track up to six people simultaneously, including two active users (Figure 2.15). The system can recognize 31 body parts per user, which allows for the detection of gestures of a certain complexity. Since launching the device, Microsoft has revealed plans to integrate Kinect with Windows (Figure 2.14) (Anthony 2010), and has released a beta version of its software development kit to the public (Microsoft 2011) enabling developers to create Windows applications that target the Kinect device. This differentiates Kinect from its competitors developed by Sony and Nintendo.

3 System Design

In this chapter, the choices dealing with the definition and the design of DETI Interact will be detailed. The content to be presented by the system will be specified, as well some requirements to present it. This will also lead to the evaluation of development platforms. Finally, the system architecture will be explained.

3.1 Introduction

The Department of Electronics, Telecommunications, and Informatics (DETI) of the University of Aveiro has, in its lobby, three large screens. These screens display a website that contains information related to the department, specifically the most recent news, and a list of all the lecturers in the department. The information contained on these webpages is stored in a server running the DSD Platform (*Distribuição de Serviço Docente*).

DETI Interact intends to replace the existing system by displaying additional information and allowing users to interact with it, enabling navigation and selection operations in the system. Interaction with the system will be done via mobile devices powered by the Android operating system. Communication between devices must be achieved seamlessly by the system on behalf of the user.

In order to conduct a better evaluation of the system, interaction with the system with a mobile device will be compared with interaction with a touchscreen, as well as a mouse. To ensure some fairness in the comparison, the system will include different types of content and methods to interact with it that better leverage the advantages and disadvantages of each device. This content includes information from the university's servers (regarding lecturers and timetables), as well as ma-

nipulation of a map from Google Earth, and three-dimensional content. Each of these sources of data has its own method of integration with the system, such as a dedicated API (Application Programming Interface) or a series of Web Services. This requires that each source be studied as to how it can be included in the system, which will in turn affect the development environment, since some APIs may be exclusive to a certain platform or programming language.

3.2 System requirements

The selected content to be presented on the initial version of DETI Interact includes content from the department's DSD Platform, as well as a map from Google Earth, and three-dimensional content. The DSD platform holds information relevant to the department, thus it will be the source for this data. From the DSD server information regarding the department's teachers and timetables will be gathered. Google Earth offers an intuitive way to use the touchscreen on the mobile device, via gestures like scrolling or pinch to zoom, making it a fitting addition to the system. The use of the digital compass and the accelerometer on the mobile device brings an advantage to the presentation and manipulation of three-dimensional content making it another suitable addition.

The DSD Platform, Google Earth, and existing 3D APIs must be studied to determine how these can be implemented in DETI Interact.

3.2.1 DSD Platform

The DSD Platform (Campos 2008) is a system that allows lecturers and students to perform operations related to the academic year. It is actively used every year by students on situations such as checking timetables, and enrolling in subjects. Students can also make choices regarding dissertation proposals, which are added to the system by the lecturers. It is, essentially, a management system for the department.

The system consists of a server computer running a Microsoft IIS (Internet Information Services) web server, which is hosting an ASP.NET web application that allows remote access to the system. The main webpage for this application can be seen in Figure 3.1. It also contains a SQL database managed by Microsoft SQL Server. The database holds all the data relevant to the department such as lists of students and lecturers, all the lectured courses along with the corresponding year and semester, class times and classroom occupation, etc.

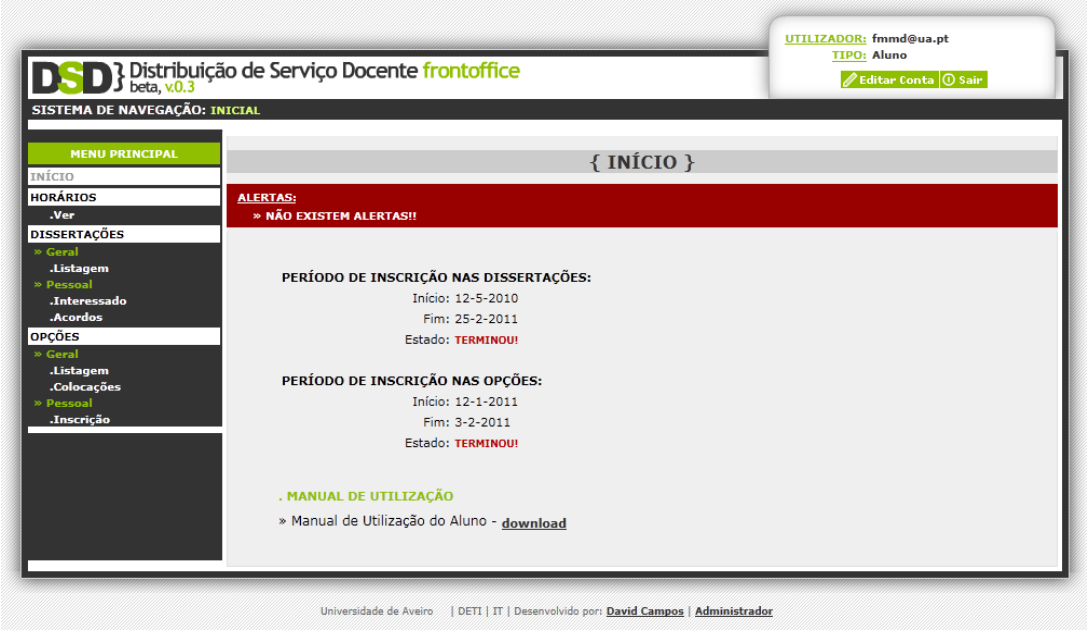

Figure 3.1 – The DSD Platform's web page.

Some of the information in the database can be accessed via a series of Web Services published by the ASP.NET Web Application. A Web Service is, as defined by the W3C, "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards" (W3C 2004).

Web services allow programmers to easily access the DSD platform's API, meaning that DETI Interact can collect and handle data from the server. The existing Web Services are quite limited in what concerns to the type and amount of information they return, but they provide enough data for some basic operations. The data returned can be used, for example, to recreate the existing list of teachers, as well as evaluate classroom occupation, generate timetables, etc. Furthermore, should these Web Services be insufficient, new ones can later be developed if the need arises. The list of available Web Services is as follows:

- getAreas();
- getAulas();
- getCursos();
- getDepartamentos();
- getDisciplinas();
- getDisciplinasAno();
- getDocentes();
- getDocentesAno();

- getSalas();
- getSemestres();
- getSubAreas();
- getTiposSala();
- getTurmas();
- getTurmasHora();
- getWishList();

3.2.2 Google Earth

Google Earth (Google 2005) is a virtual globe application, with mapping and geographical features. It maps the earth's surface by overlaying images obtained from satellite imagery and aerial photography. The product was initially released as a standalone, downloadable application. However, it has since become accessible via a web browser plugin.

Introducing Google Earth to DETI Interact provides a way of comparing different methods of interaction. Given its graphical user interface, it constitutes a suitable choice to compare interaction with mouse and keyboard, with interaction using a touch screen on a mobile device. Google provides APIs for developers to use its mapping services, making integration of the Google Earth plugin possible. This API, however, is only accessible through Javascript, which is intended for use on web applications. Including Google Earth on the system involves finding a way to parse HTML and Javascript and displaying the results within the application. This can be done by either accessing Web Browser controls or using libraries on the OS for that purpose.

3.2.3 3D Content

Manipulation of three-dimensional content represents another appropriate choice for the comparison of different interaction devices. The use of compass or accelerometer on mobile devices can be compared with the use of the mouse and keyboard, or a touchscreen, on a computer.

There are numerous ways to display three-dimensional content on a computer, including OpenGL, DirectX, VTK, and XNA. These graphical APIs are used to produce standalone applications, but can also be used to display three-dimensional content within applications, albeit with some minor adjustments. The use of any graphics API will depend on the development platform: OpenGL and VTK are cross-platform, DirectX is restricted to C++ development on Windows, although there are managed wrappers for .NET, and XNA is a Windows exclusive as well.

In order to use any of these APIs to render a scene within a desktop application it is necessary to either use an available wrapper, which allows the developer to create a user interface around the three-dimensional content, or redirect the target render buffer to a different container (e.g. an image), and display this container in the application.

3.2.4 Mobile Devices

In order to target mobile devices it is necessary to decide whether to develop a web application, enabling all the mobile platforms to interact with the system, or to develop a native application, which targets a specific platform.

The ideal solution would be to develop a Web Application for mobile devices instead of a native application since it would be able to target any mobile device with a web browser, regardless of operating system type or version. It also provides better control over the application: there is only one application to develop and update, and every user will always be using the latest, most up-to-date version. However, the problem with web application would be accessing the device's hardware (Fling 2009). For DETI Interact, apart from the touchscreen, it is also useful to have access to hardware such as the accelerometer and Bluetooth adapter. In addition, access to the camera, GPS, storage, etc., might be necessary in future versions of the application.

Another solution for distributing for multiple platforms would be using PhoneGap (PhoneGap 2008). This mobile framework currently allows development for six different mobile platforms: Apple iOS, Google Android, Blackberry OS, Palm webOS, Windows Mobile, and Symbian. It provides access to hardware such as the accelerometer, camera, compass, GPS, and storage, but still offers no access to the Bluetooth adapter. Navigation in the applications built using PhoneGap also are not consistent with the operating system, for example, pressing the 'back' button on an Android device exits the application, instead of navigating back to a previous screen.

Since the mobile application for DETI Interact requires access to hardware unavailable when developing for both the mobile web and PhoneGap, the remaining solution is to develop a native application. Given that the Global market share for the Android platform has revealed a rapid growth and a high adoption ration, quickly becoming the leading mobile platform (DailyTech 2011) this was the targeted platform for the initial version of DETI Interact. In addition, there was already Android-powered hardware from previous studies, available to be used for this system.

Developing for Android is done via a provided SDK that allows access to all the required hardware, which is unavailable or limited in other platforms such as iOS or Windows Phone 7. Other mobile platforms can easily be addressed at a later stage.

3.3 Development Environment

With the specification of which content is to be presented, steps can be taken into the development of DETI Interact. Given that the DSD Platform is developed using the .NET framework, and Android devices require the use of the Java platform, the use of either of these environments to develop DETI Interact would be beneficial given the vast amount of existing libraries for each platform. Content such as Google Earth and 3D scenes can be easily included in an application developed in any language.

Since the target for the mobile application is an Android device, the Java platform could also be used to develop the application that will be running on the large screens. One big advantage of using Java would be its platform-independence, since the Java Virtual Machine is deployable to most known platforms. Being such a widespread platform, a vast amount of libraries, wrappers, and documentation is also available. Additionally, the Java Class Library includes a Bluetooth API, which would be useful for communication between computers and Android devices. On the other hand, while its performance has improved substantially over time, Java still displays poor performance in some scenarios, as well as a high consumption of system resources (Mihailescu 2010). Furthermore, regarding security, Java has become the most common target for malware, surpassing Adobe Flash and Adobe Reader (Cull 2010). This could represent a security issue since the system is to be deployed at a public location.

An alternative for the Java platform is the .NET Framework (Microsoft 2009). While it cannot match Java in platform independence, .NET still covers a wide range of platforms thanks to the Mono Project that implemented its own version of the Runtime; Mono's implementation of the framework, however, lags behind Microsoft's in some areas. The framework developed over time and now includes features unavailable on the Java platform such as Language Integrated Query, for data processing, and Windows Presentation Foundation, an innovative tool used for user experience and user interface design. In terms of performance, while .NET (and managed applications in general) will probably never be as fast and memory efficient as their native counterparts, applications tend to be faster, due to its Just-In-Time Compiler, and less resource intensive, due to its Garbage Col-

lector (Mihailescu 2010). In Addition, .NET assemblies also take advantage of Windows' Data Execution Prevention and Address Space Layout Randomization, which improves the overall security of the system (Richter 2010).

While communication via web services is independent of platform or language, consuming .NET Web Services from a different platform, such as Java, is not as streamlined as doing so from within the .NET platform. Google Earth and 3D content can be implemented in either platform. Furthermore, communication between the Android device and the desktop application via Bluetooth is done through the exchange of packets of data, such as OBEX messages, which is independent of platform or language. Taking in to account the pros and cons of each alternative, the chosen development environment for the computer application was the .NET Framework.

3.4 Development Tools

3.4.1 Software

In order to develop DETI Interact, two applications must be developed: a desktop application, in .NET, and an Android application in Java. Various tools were used to assist and speed up the development of these applications.

3.4.1.1 Microsoft Visual Studio 2010

Since the system would be built on the .NET platform, using Visual Studio was an understandable choice. Microsoft Visual Studio 2010 is the most recent version of this IDE (Integrated Development Environment). It ensures quality code throughout the entire application lifecycle, from design to deployment, and assists in the development of applications for Windows, Windows Phone, SharePoint, Web, Cloud, etc.

Visual Studio was used to develop the portion of the system that ran on the computers at the lobby of the department. The application along with some addition libraries was developed in C#, and the user interface was built with XAML, used by Windows Presentation Foundation and Silverlight.

3.4.1.2 Microsoft Expression Blend 4

Expression Blend is a user interface design tool belonging to Microsoft's Expression Studio suite. It is an interactive front-end for designing XAML-based interfaces for Windows Presentation Foundation, Silverlight, and Windows Phone applications (Microsoft 2011). Blend allows developers to focus on the user interface design of the applications, without affecting the business login layer. It promotes the usage of the MVVM architectural pattern.

3.4.1.3 Eclipse

Eclipse is a multi-language, multi-platform software development environment. It consists of an Integrated Development Environment and an extensible plug-in system. This plug-in system allowed for the development of the Android Development Tools (ADT) plug-in which includes a device emulator, and tools for debugging, and memory and performance profiling. It is the IDE recommended by Google, and was used for the development of the Android application.

3.4.2 Hardware

DETI interact is expected to run on a computer connected to a large display as well as a computer equipped with a touchscreen. These are both Asus machines: an Eee Box, connected to the large display, and an Eee Top. The mobile application that will interact with the system will be tested on a provided Motorola Milestone.

3.4.2.1 Asus Eee Box

The Asus Eee Box is a small form-factor desktop computer. It has a low-end processor and integrated graphics card, which allows it to consume very little power. This, however, can have a negative impact on the performance of the application, since the system will deal with very large sets of data, as well as three-dimensional content.

3.4.2.2 Asus Eee Top

The Asus Eee Top is a computer similar to the Eee Box line, but its hardware is built into a touch-sensitive screen. In addition to the low-end hardware, the screen only supports single-point interaction, which will limit the set of gestures that can be used as well as affect the overall user experience.

3.4.2.3 Motorola Milestone

The Motorola Milestone is a quad-band version of the Motorola Droid, an Android-powered smartphone. The hardware is similar between the two, with only minor differences in internal memory and installed applications. It was the device used to test the developed mobile application, and was also used for the tests performed with users. During the time the DETI Interact was deployed to the computer in the lobby, users were able to install the Android application in their personal devices.

3.5 DETI Interact Design

With all the system requirements taken care of, a development environment chosen, and all the hardware detailed, the system architecture as well as some design and development considerations will be described.

3.5.1 Architecture

As discussed previously, DETI Interact will consist of two applications. The first application will run on the desktop computers connected to the large displays. This application will contain the content gathered from the DSD server, the map from Google Earth, and a viewer for 3D content. To allow interaction from Bluetooth devices, a Bluetooth socket will be waiting to acquire a device. The desktop application will interpret data packets sent by the mobile device in order to perform operations such as changing the displayed content. While no device is connected to the system, the desktop application will be able to enter a demonstration mode, cycling through all the available content.

The second application will run on an Android mobile device. It will observe gestures drawn on the touchscreen, as well as the values from the digital compass or accelerometer, and send this data as a structured packet via Bluetooth to the desktop application. A diagram of the system architecture can be seen in Figure 3.2.

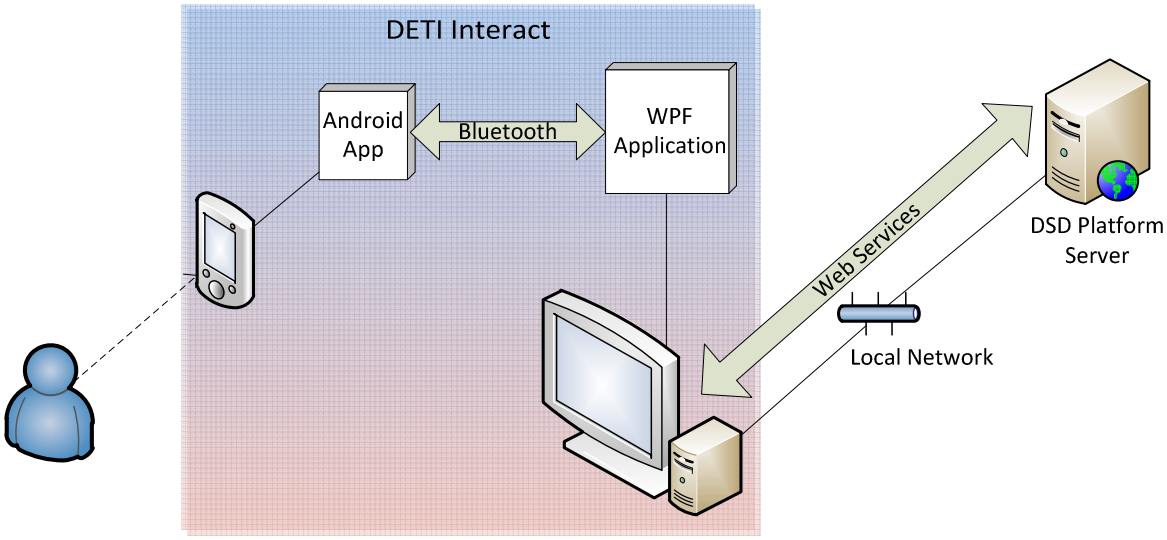

Figure 3.2 - DETI Interact architecture

3.5.2 Desktop Application – .NET Framework

As discussed previously, the .NET Framework was chosen as the development environment for DETI Interact. The mobile application, developed for Android devices, will be written in Java, and the desktop application, which will run on the computers on the department, will be written in C#.

The .NET framework is a programming environment developed by Microsoft for building, deploying, and running applications and services that use the .NET technologies. The framework consists of two main components: the Common Language Runtime (CLR) and the Framework Class Library (FCL). The CLR is a runtime that contains features such as memory management, assembly loading, security, exception handling, and thread synchronization, and is usable by various programming languages. Currently dozens of languages can target the runtime of which the most widely used is C#. This was the programming language used in the development of DETI Interact. It also includes features that are not available in other platforms but are useful for this system, such as LINQ and WPF.

LINQ – Language Integrated Query

Using .NET benefits DETI Interact when it comes to handling data from the DSD database due to the inclusion of LINQ. This feature adds native querying capabilities to .NET languages meaning

that the programmer can write code similar to SQL queries in order to parse data, instead of looping through all the results returned from the web services. This can speed up certain portions of the code, ensuring a more responsive application.

WPF – Windows Presentation Foundation

Another noteworthy addition to the .NET Framework is Windows Presentation Foundation (WPF). Microsoft describes WPF as being "a next-generation presentation system for building Windows client applications with visually stunning user experiences" (Microsoft 2010). Instead of depending on the older GDI subsystem like Windows Forms or MFC, WPF uses DirectX, which allows Windows to offload some graphics tasks to the GPU, enabling the CPU to work on tasks that are more significant. WPF also introduced a new architectural pattern: Model-View-ViewModel (MVVM). This pattern was created to make use of specific features in WPF and Silverlight, separating the View layer from the rest of the pattern. This separation of layers means that the business logic of the application is completely independent of the user interface.

3.5.3 Mobile Application – Android

As described on 3.2.4, the targeted mobile platform was the Android platform. Android is a software stack for mobile devices that includes an operating system, middleware and key applications (Google 2011). An SDK (Software Development Kit) is provided with the tools and APIs necessary to develop application on the Android platform.

Development for Android is done in Java, however, there is no Java Virtual Machine, instead the code targets the Dalvik virtual machine, which was specifically designed for Android and optimized for battery-powered devices with limited memory and CPU. The system integrates a web browser based on the WebKit engine, SQLite for data storage, and support for common audio, video and still image formats. It also features GSM telephony, Bluetooth, EDGE, 3G, WiFi, Camera, GPS, compass and accelerometer. All of these features are available in the platform's SDK.

The provided SDK allows developer access to the gestures drawn on the touchscreen, data from the compass and accelerometer, and the Bluetooth adapter. These are the necessary features for the system to work properly.

3.5.4 User Interface and User Experience

Designing a good user interface for the system and planning a good user experience are crucial to increase interest and to encourage users to use the system. The user interface is what allows the user to manipulate the application. It must provide effective ways for a user to operate and control the system. This should be done while providing a good user experience. User experience is a subjective concept since it varies between users. The system must be easy and designed in a way that the user does not feel the using the system is a chore. The user must feel that he/she knows how to operate the system at first contact, and can anticipate how the system will respond to his input.

There will be two very distinct screens in use: the large screen, and the mobile screen. This means that some care must be taken in how the screen will be used. Depending on how the screens display information, using both screens may cause user confusion. In order to avoid having the user diverting attention between the large screen and the mobile screen, the chosen approach was to have all the content presented on the large screen, and use the mobile device solely for interaction. This will allow the user to focus on the large screen, and forget about the mobile device, reducing the cognitive load on operating the mobile device correctly, in an attempt to resemble what happens with the computer mouse nowadays.

One other very important detail is how the information will be displayed on the screen. The available space must be used properly and it must provide a decent usage experience. Several aspects must be taken into account:

- The system will also be deployed to a touchscreen computer, which means that interaction with content on the screen must be finger-friendly;
- The large display is fixed on a wall, meaning that users will stand further away as they would from a regular computer, hence, content must be displayed in a larger format;
- Both screens have a widescreen format (with a 16:9 image ratio), with their width being almost twice as large as their height. Given the more limited height, screen space would be better used, for example, by stacking controls in a column rather than in a row (Figure 3.3), this, however, limits the amount of controls that can be stacked;

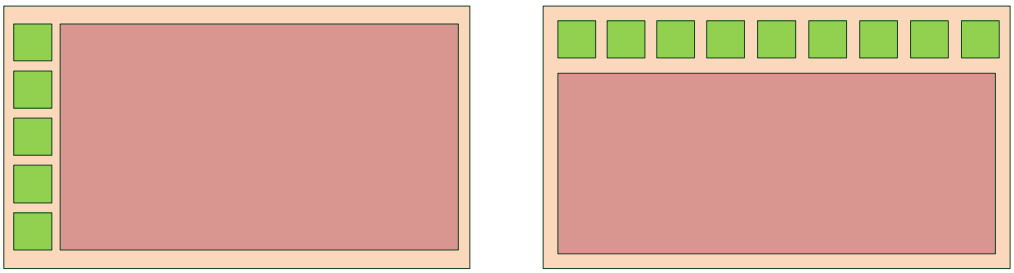

Figure 3.3 – Two distinct ways of using available space on a widescreen display.

- The screens will be manipulated by either mobile device or a touch screen, therefore the content, and interaction with it, must support a limited set of gestures.

Figure 3.4 - The original Xbox 360 dashboard (Thurrott 2010)

There are numerous ways of displaying content properly, each with their advantages. Media centres and videogame systems provide great user interfaces, directed for controllers with limited functionality namely directional and selection buttons. These systems provide a good starting point in the development of the user interface. The original dashboard included with the Xbox 360 video game system (Figure 3.4) provides a large amount of screen space for information, with different panes that can be used to switch context. This dashboard is derived from the UI used in Windows Media Centre, and newer versions of this dashboard have eventually blended with it. The concept behind these user interfaces has been in constant evolution for over a decade culminating in devices such as the Zune player and the Windows Phone (Thurrot 2010).

The original Xbox dashboard served as inspiration in the design of the user interface for the desktop application of DETI Interact. With information arranged in a similar manner a greater portion of screen can be used to display information, while the edges show which other panes area accessible.

3.6 Conclusion

In order to make DETI Interact a system appealing to the user, a series of features were chosen to integrate it. Communication with the DSD platform is required to provide content relevant to the department, and can be used to populate the system with a list of teachers, timetables, etc. In addition, a Google Earth plugin and the exhibition of three-dimensional content would make the system more engaging, as well as providing interesting methods for interaction with a mobile device.

The system will include two applications, a desktop application, which will be developed in .Net, for its performance, security, and included features, while the mobile application will be developed in Android, using Java, since this is the standard approach.

4 Development

In this chapter, both the mobile and the desktop application will be described in detail. In addition, some implementation details will also be presented.

4.1 Introduction

As mentioned, DETI interact is a system that uses two different applications: a mobile application, running on an Android device, and a desktop application running on a computer, which displays information on a large screen.

The Android application is tasked with collecting different gestures performed by the user. With no information being displayed there, the user can focus on the large screen, and use the mobile device merely as another peripheral.

The desktop application is responsible for displaying the information, as well as interpreting the gestures from the mobile application, and responding accordingly. This required a series of libraries to be developed. These were needed for fetching and processing the data from the DSD server, to enable the inclusion of and XNA application, and to interpret the gestures performed by the user.

4.2 Android Application

To interact with the desktop application a mobile application running on the Android operating system was developed. This application connects to a DETI Interact computer and transmits data through a Bluetooth connection. A Bluetooth service was developed to perform all Bluetooth-

related operations. In addition, the Android SDK supplies a `GestureDetector` class that was used for the detection of gestures. The values from the digital compass are also sent via Bluetooth to the computer.

4.2.1 Bluetooth Service

The `BluetoothService` class controls the communication with the desktop computer. To perform the connection, the Bluetooth MAC address of the computer must be known. This information is gathered when the devices are paired (as stated in 4.3.2.3, Android's Bluetooth API enforces device pairing). When the application is launched, a list of connectable Bluetooth devices is displayed. Once the user chooses a device, its address is passed to the `BluetoothService` class, which then tries to connect to it. If the connection succeeds, the input and output streams are initialized, enabling the application to read and write from them.

A `write()` method is supplied by this service that can be used to place an array of data into the output stream of the Bluetooth connection. The streams between two Bluetooth (or Serial) endpoints can be written or read at any time, by different clients. If two clients were to write at the same time, whoever reads the stream afterwards will find a mixture of both messages. In order to avoid this situation, the provided `write()` method only allows one message to be written at any time. Whenever the system needs to write some data to the stream, the `write()` method blocks any further calls, in order to avoid conflicts. While the method is blocked, any awaiting messages are discarded, since the system is expected to respond to user input in real time, and answering these messages would result in the system responding with some lag.

4.2.2 Main Application

The main activity for the Android application requests a Bluetooth connection to the aforementioned service. The information collected by the sensors and sent to the Bluetooth services is placed in a string, with a predetermined size and structure. This structure, shown below, identifies the gesture, as well as three distinct fields, for the X, Y, and Z components of a scroll or fling, or the Yaw, Pitch and Roll components from the device's orientation. Unused fields are left with a null value.

Bytes:	0	1-2	3	4-12	13	14-22	23	24-32	33
Content:	"["	Gesture ID	":"	Value 1	":"	Value 2	":"	Value 3	"]"

This structure has a fixed size to aid the computer in the decoding. This eliminates some problems that existed on an initial version of the system where the computer was decoding messages at a slower rate than they were being sent. This caused some conflicts in the messages such as messages cut in half, or concatenated messages, which interfered with the gesture detection.

Concerning the detection of gestures on the device's touchscreen, the `GestureDetector` class offered by Android's API identifies most of the needed gestures such as Tap, Scroll, and Fling. However, the Pinch gesture, used to zoom in and out, is unavailable in the API for Android 2.1. A `ScaleGestureDetector` class was introduced in version 2.2 of the API to allow the detection of pinch to zoom gesture.

This constitutes one of the main problems of the Android platform: fragmentation. Mobile operators and OEMs (Original Equipment Manufacturers) load devices with their software, or modified firmware versions, which have to be adapted when Google launches a platform update. New features, such as the `GestureScaleDetector`, are added to these updates, and are not backwards compatible. Most devices will only receive these updates later if at all. The result is that the Android ecosystem includes devices running android versions from 1.6 up to 3.0. If an application uses a feature from the API from version 2.2, every device running Android 2.1 or under will not be able to run the application. For this reason, and since development was started when Android 2.1 was the most recent Android version, the zoom gesture had to be implemented from scratch. This is done by detecting two fingers on the device's touchscreen, and accompanying the variation in spacing between the fingers.

4.2.2.1 Supported Gestures

The set of gestures currently supported by the system include Scroll, Fling, Tap, Long Press, Rotation, and Zoom.

Scroll: scrolling (Figure 4.1) is done on the mobile device by touching a point on the device's touchscreen, and dragging the finger to another position. Any direction on the XY axis is valid and therefore transmitted to the desktop application.

Fling: the fling gesture (Figure 4.2) is a quick finger swipe, across the device's screen, in any direction. It is usually used to perform an accelerated scrolling gesture, known as Kinetic Scrolling.

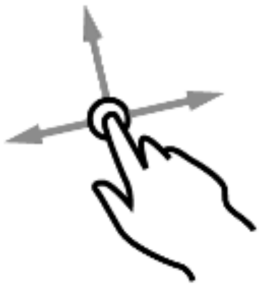

Figure 4.1 – The Scroll Gesture (Ideum 2011).

Figure 4.2 – The Fling Gesture (Ideum 2011).

Figure 4.3 – The Tap Gesture(Ideum 2011)

Tap: a tap (Figure 4.3) consists of touching the screen for a short duration of time. It is usually used to perform selections.

Long Press: a long press (Figure 4.4) is done by touching the screen for a longer period of time (about one second), which differentiates itself from the tap gesture, as well as accidental touches. This gesture is usually used to trigger the display of a context menu.

Pinch to zoom: zooming (Figure 4.5) is done placing two fingers on the touchscreen and moving them closer together or apart, to zoom out or in, respectively.

Figure 4.4 – The Long Press gesture (Ideum 2011).

Figure 4.5 – The Pinch to zoom gesture (Ideum 2011).

Figure 4.6 – The Rotation gesture.

Rotation: the values from the device's digital compass are constantly polled and sent to the desk-top application. These values are used to detect several rotation gestures (Figure 4.6).

The development of the desktop application can be targeted to take advantage of this gesture library.

4.3 Desktop Application

As discussed in section 3.2.1, the Web Services available in the DSD platform, albeit limited, are enough to list lecturers and generate schedules. These two features were chosen to integrate the first version of the application. In addition, since current mobile devices are equipped with multi-touch displays, accelerometers and digital compasses, a map and a three-dimensional object viewer were also included in the application.

During application development, attention was taken to ensure system extensibility, so that it would be easy to add, remove, or modify core features.

4.3.1 Structure

The desktop application was developed in C# using Windows Presentation Foundation (WPF) for the user interface, using the Model-View-ViewModel pattern where appropriate. This prompted for the structure shown in Figure 4.7.

The application presents a user interface with a series of pages, with different content being presented on each page. In the initial version of the application, there is a page for the list of teachers, a page for the timetables, a page for the Google Earth plugin, and a page for the 3D model viewer. There is also an additional page where help is presented to the user. Every page hosts a WPF control, which is tasked with displaying the specific content, and offers several interaction possibilities. This is achieved by using the developed `DetiControl`. This control provides a series of properties, methods, and events so that the application can control its operation. By adding a `DetiControl` to the application, a new page is automatically created on the UI and it will operate in the same way as any existing control.

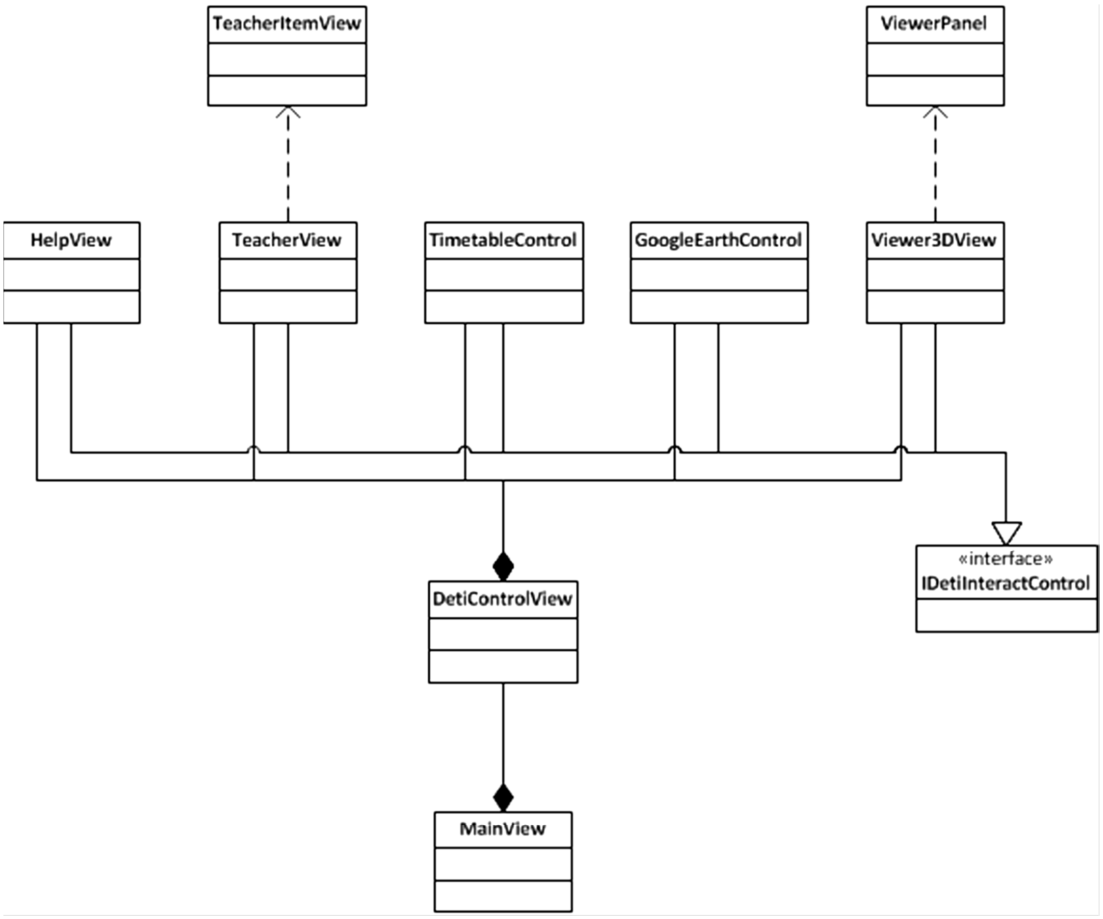

Figure 4.7 – Structure of the main application.

Each page must also display some content. In order to do so, the `IDetiInteractControl` interface is supplied. This interface provides methods that allow the system to perform operations such as triggering demonstration mode, and reacting to user input. Using these two components, all the developer has to do to add new content to the application is:

- Create a new WPF control to host the desired content.
- Implement the `IDetiInteractControl` interface. The control should have its own animation routine, and should respond to user input.
- On the main UI file (`MainViewModel`) used the provided method to add the control, and the title to be displayed on the page:
    - `AddWindowControl(new DetiControlView(new WPFControl(), "<Control Name>"));`

The developed control will have been added to the user interface, a new page will have been created, and the control will respond to user input and animation events.

4.3.2 Developed libraries

In order to offer the features discussed in Chapter 3, a set of auxiliary libraries was developed to provide the necessary content, namely, the DSD Provider library, and the Model Viewer XNA library. The DSD Provider communicates with the DSD server to collect the necessary information to present the lecturer list and the timetables, and the Model Viewer XNA library consists of an XNA application, which will render the models. Including a Google Earth plugin does not require the use of an auxiliary library, so it can be done within the main application.

In addition, two separate libraries were also developed, one that allows interaction from remote devices, the Control library, and one used to perform logging of events, the Logger library.

4.3.2.1 DSD Provider

As discussed in Chapter 3.2, DETI Interact needs data that is stored on the DSD platform, which is accessible via web services. Accessing web services, however, can take some time due to communication delays. Furthermore, processing the vast amount of data returned can also be time-consuming. These reasons prompted for the creation of the `DSDProvider` class library (Figure 4.8).

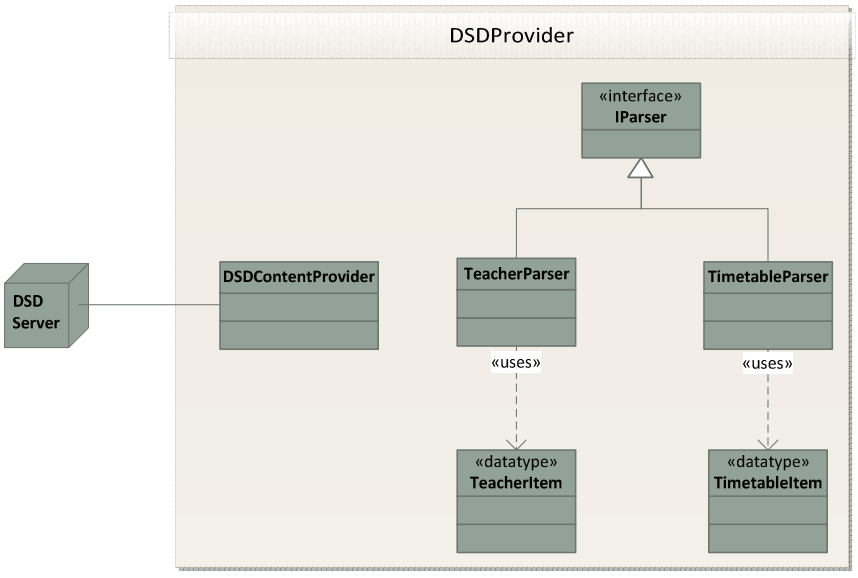

Figure 4.8 – The `DetiInteract.DSDProvider` class library.

This library is composed of two parts: the `DSDContentProvider` class, tasked with collecting all the data, and the Parsers, that work on the collected data.

4.3.2.1.1 DSDContentProvider

The `DSDContentProvider` class is responsible for accessing the Web Services and retrieving information from the server.

In order to use a Web service in a .NET project a proxy class must be created and compiled, this class is generated from a reference to the Web Service, typically the WSDL file. Using Visual Studio, this is achieved by adding a Service Reference to the project and assigning the address to the web services. Once the reference is added, the proxy class is generated automatically. The method for consuming a web service would be similar to that of calling a method from an external library.

4.3.2.1.2 Parsers

The Parser classes are functional components that work with the data provided by the `DSDContentProvider` class. Each parser processes a specific set of data and sends the result back to the application.

To ease system extensibility, an `IParser` interface was created. This interface lists the methods that every parser should implement. This allows the developer to determine how the parser will perform its functions and what data will be processed. The interface also publishes an event that can be used to notify the application of changes in the parser (such as when new data is available).

In the present state of the DETI Interact system, two distinct parsers are implemented: the `TeacherParser` and the `TimetableParser`.

In terms of performance, since calling Web services usually takes up more time than calling an internal operation, `BackgroundWorker` instances were created in each parser to collect and process the information. A `BackgroundWorker` allows an operation to be run on a separate, dedicated thread. Time-consuming operations which the parsers are likely to perform can make it seem as though the UI has stopped responding for a period of time, running it on a separate thread maintains responsiveness, improving user experience.

TeacherParser

The system that was previously running in the lobby contained a web page that displayed information regarding all the teachers of the department. The page would automatically scroll to different positions so that all would be shown. Information about each teacher was organized in a business card fashion with his or her name, photo, web page, office door number, email, and phone

number. This behaviour was to be reproduced in DETI Interact so this information had to be extracted from the list of teachers that the `DSDContentProvider` had fetched.

Extracting the required data from the `DSDContentProvider` is a simple operation although it requires looping through all the elements in the list. The `BackgroundWorker` loops through the list and extracts a teacher record, it then sends this record the `ProgressChanged` handler, which collects the required data and places it in a data structure. This structure is then sent to the UI thread via the existing event.

TimetableParser

The web page for the DSD platform (http://dsd.av.it.pt) allows the user to check timetables for each course and grade at the department, this feature was chosen to integrate the initial version of the DETI Interact system.

Creating a visual representation of a timetable requires a more careful approach since the Web services can only return textual information such as a list of all the subjects, classes, and classrooms, class times, etc., which constitutes an immense amount of data to be processed. In order to determine when a certain class takes place, it is necessary to navigate through six tables, which can be an arduous task for the CPU. Since the UI thread has to be able to quickly draw a timetable, most of the work will have to be done in the parser.

After accessing all the lists in the `DSDContentProvider`, the parser must extract the required data in order to create every item for the timetables. This data will be placed into a data structure (the `TimetableItem` class) that will be sent to the UI thread for drawing. The classic approach of looping through all the lists attempting to match IDs would result in an algorithm with a complexity of $O(N^6)$. This is where LINQ becomes a very useful asset. The querying capabilities of LINQ enable lists to be joined making it only necessary to loop through the resulting list once, lowering to complexity of the algorithm down to O(6N). This resulting list contains all the `TimetableItem` instances for a given year and course combination, and these can now be sent to the UI thread via the existing event.

4.3.2.2 XNA Model Viewer

One of the features previously discussed had the application display three-dimensional content. This would be used as a virtual tour of the department, allowing the user to visualize models or navigate in a virtual representation of the department.

There are numerous ways to display three-dimensional content on a computer, including OpenGL, DirectX, VTK, XNA, and WPF. Ideally, this 3D content would be rendered within the application instead of a standalone application. For this scenario, the first choice would be to use the 3D API included in WPF, as it would render content directly in the application window. However, this API is not as full-featured as OpenGL, DirectX, or XNA, meaning it would not scale as well with larger, more detailed scenes. The alternative APIs render 3D content to a specific buffer, such as the full screen buffer, or a given window handle, so it is required to replace this target buffer with a certain portion of the window.

Previously, rendering 3D content within an application required a Windows Forms control, which could be hosted within a WPF application. While this worked without problems, performance was severely affected. With version 3.5 of WPF the D3DImage `ImageSource` was introduced, which can be used as a render target for DirectX. Since XNA is a managed library built on top of DirectX, this was the library of choice for the Model Viewer.

The Model Viewer library is a standard XNA library project. It includes a series of 3D Models in FBX (Autodesk) or X (DirectX) formats, which are loaded and rendered by request of the UI thread. It also provides basic camera interaction.

4.3.2.3 Control

One of the main objectives of the DETI Interact system is the ability to control it via an Android mobile device. This is possible through the `DetiInteract`.Control library. This library is used to receive data from the Android device and interpret a certain gesture; it will then notify the UI thread of the gesture so that the application may perform accordingly. This implementation allows the application to decide how to deal with each gesture; it also allows the library to be reused in a different scenario.

This library contains a Controller class, and the `CommHandler` classes. In this version of the system, the chosen communication interface was Bluetooth, resulting in the creation of the `Bluetooth-`

`CommHandler`. Other `CommHandler` classes can be added as required, ensuring reusability and extensibility of the library.

4.3.2.3.1 BluetoothCommHandler

The `BluetoothCommHandler` class manages communication between Bluetooth enabled mobile devices. While the system is targeting Android devices in particular, any Bluetooth device will work with this class. All Bluetooth communication is handled by In The Hand's 32Feet.NET library (Foot). This library creates a higher level of abstraction on top of the Win32 API.

Setting up communication between Bluetooth devices is a usually straightforward task but Android's Bluetooth API requires those devices to be paired. Since the system is to be deployed to computers connected to a large display, with input methods inaccessible to the user, pairing must be done automatically and seamlessly. This presented a problem during development since different versions of Windows supported different Bluetooth specifications. Windows Vista and Windows 7 both support Bluetooth v2.1, which automatically performs Simple Secure Pairing with devices that support it. With Simple Secure Pairing, both devices agree on a sequence of characters, which the user then has to confirm on both devices. This feature overrode any attempt to pair the devices within the application, requiring user input on the Operating System level.

Windows XP did not present this problem, other solutions would include using and older Bluetooth adapter on the Windows machine, or using the Win32 API directly. The Android SDK version 2.3.3 features an API for insecure Bluetooth socket connections (CellPhonesMarket 2011), which will avoid this issue

To perform device pairing within the application a callback must be set to answer pairing requests. This callback only has to define the pin that the mobile devices must use to pair. This way pairing will be done and the Operating System will not interfere in the process.

Communication between devices using Bluetooth is similar to communication using a Serial interface (Bluetooth is essentially a wireless implementation of Serial communication). The devices must know the MAC address for each other, open a socket, and write into or read from that socket. Since the DETI Interact system will be deployed in a public setting, many Bluetooth-enabled devices will be in range, and polling every device would not be a very practical approach, instead the system will be constantly waiting for a device to request a connection. After accepting the connection, the system asks the device for the shared Stream. This stream will then be used to read the packets

sent from the mobile device that contain data pertaining to the gestures performed on the device. Since waiting for a connection request from a device is a blocking operation, and maintaining a continuous connection with a device is time consuming, the code must be run on a separate thread.

The data read from the stream encodes data from the mobile device. This data consists of a gesture identifier and a set of values corresponding to the gesture (for example, the X and Y speed of a fling); however it can also notify when a device is about to disconnect. This resets the listening process, leaving the system to wait for a new connection. Gestures are reported to the Controller via an event.

4.3.2.3.2 Controller

The `BluetoothCommHandler` class handles communication with remote devices via a Bluetooth connection and then report to the Controller class. Additional classes can be added to this library as needed to allow for different methods of communication. The `Controller` class' purpose is to manage the existing `CommHandler` classes. This class interprets that data received by the `CommHandlers` and notifies the UI thread via an event with details regarding to gestures performed on the mobile device.

4.3.3 Displayed Content

Various controls were included in the initial version of DETI Interact, namely a control that displays a list of teachers, a control that displays the students' timetables, a control that allows navigation in an interactive map, and a control that enables the user to manipulate a three-dimensional object. An additional page was added to the application displaying helpful information to the user regarding system usage.

As specified in the previous chapter, the displayed content will be organized in several pages, in a tabbed interface, with vertically oriented tabs arranged along the edges of the screen. Given the available set of gestures, it was decided that the touch-based gestures would be used to interact with the displayed content, while the orientation-based gestures would be uses to alternate between pages. The reason behind this choice was that the use of the digital compass would place the user under a higher cognitive load, so it would be more appropriate to leave it to secondary actions. This choice was later confirmed with user testing, which can be seen in Chapter 5. Interac-

tion with the content displayed in the pages is implemented by each page, which will be detailed below.

4.3.3.1 Teacher Page

As stated in 3.1, the previous system displayed a list of information regarding the teachers of the department. Information about each teacher was organized in a business card fashion and displayed in a matrix on a web page. This page would automatically scroll to pre-set locations so that all teachers would be displayed. The teachers' page in the new system implements a similar behaviour.

The teacher page consists of a Teacher Control, which hosts a list of `TeacherItemControls`. Both the controls are built using the MVVM pattern, taking advantage of the databinding capabilities of WPF. The `TeacherItemControl` was developed to house the information for each teacher. The ViewModel for the `TeacherControl` is responsible for dealing with the data. It starts by creating an instance of the `TeacherParser`, which returns a list of `TeacherItem` instances. These are used to create the `TeacherItemView` controls, which are placed into an `ObservableCollection`. WPFs databinding engine ensures that the contents of the listbox on the page match the contents of the collection at all times.

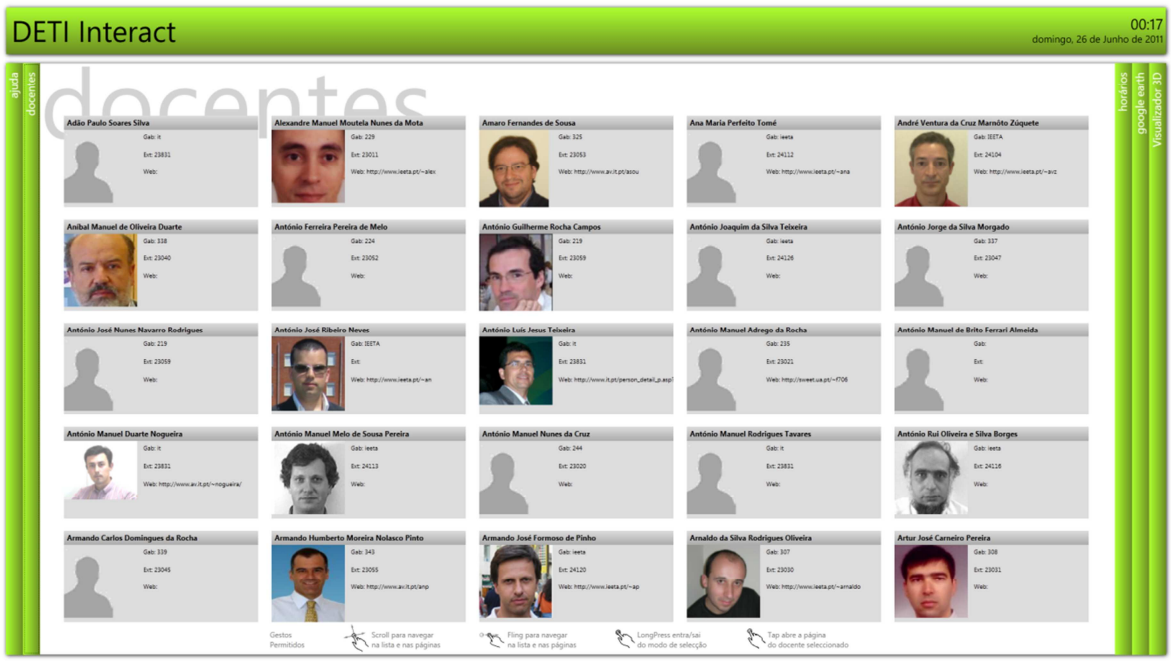

Figure 4.9 – The Teachers' page.

This control is configured to use all the available screen space, regardless of resolution, meaning that more information will be visible in the larger screens whilst remaining easy to read. For instance, a 1080p resolution screen will display five columns of teachers, while a 720p resolution screen will only display three columns. Figure 4.9 shows the final look for the teachers' page.

In addition to displaying the list of teachers, this control also displays the web page of any selected teacher if the URL is in the database. For the page to be displayed, a Web Browser control is overlaid on the list of teachers (Figure 4.10). When selecting a teacher, the address to his webpage is passed to the browser and the page is loaded. The Web Browser control was configured to prevent new windows from being opened, which would cause the application to lose focus, and become concealed by the newly opened window.

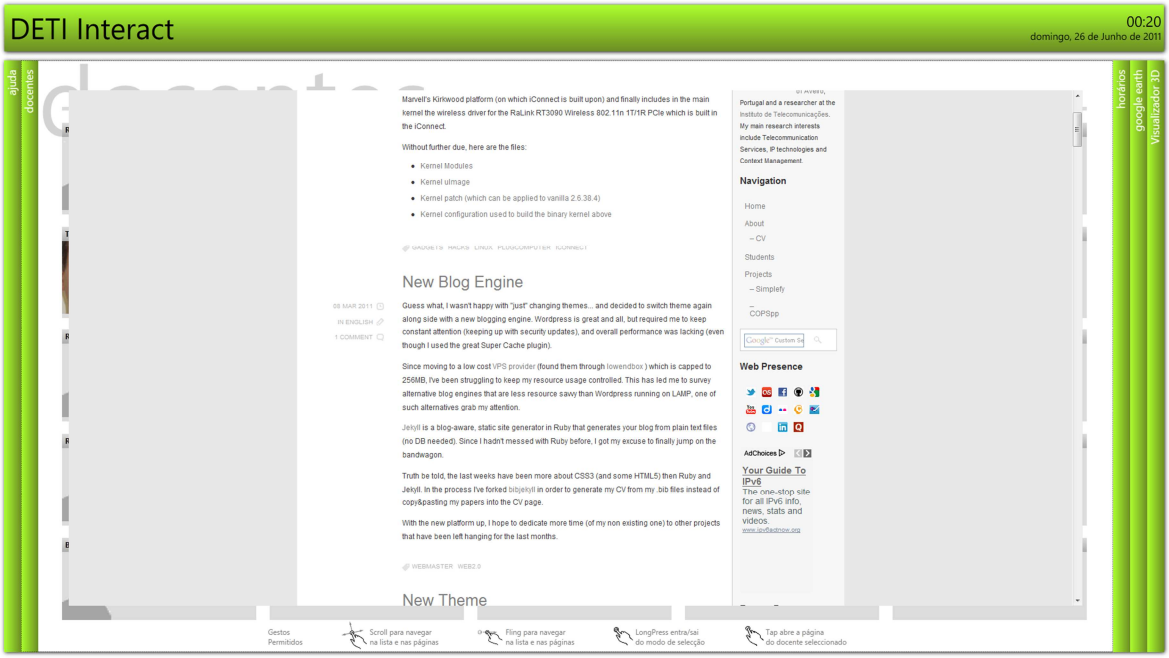

Figure 4.10 – The Web browser, visible after a user selects a teacher.

4.3.3.1.1 Interaction

To interact with this control, several gestures are provided. The users can scroll vertically along the list of teachers, or use a vertical fling gesture to trigger kinetic scrolling. In order to browse the teacher's webpage, the user must enter selection mode. This is done via a `LongPress` gesture. This gesture is used to enter and exit selection mode. Upon entering, the teacher closest to the centre of the page is automatically highlighted. The user can then use flings, in any direction, to select any

adjacent teacher. Once the desired teacher is selected, a simple Tap gesture will launch the web page.

Once the Web Browser control is visible and the webpage loaded, the user can scroll or fling, in any direction, to view the full extent of the webpage, in a manner similar to what is done in modern mobile web browsers. To leave the browser and return to the teacher list, the Long Press gesture is used once again.

4.3.3.2 Timetable Control

The previous chapter specified how the `TeacherParser` was used in creating the teachers' page; a similar approach was taken to develop the timetables' page, using the `TimetableParser`. The parser was built in a way that frees up the control of the greatest part of the work; however, the control still has to place every item in its correct place, in order to create a readable timetable.

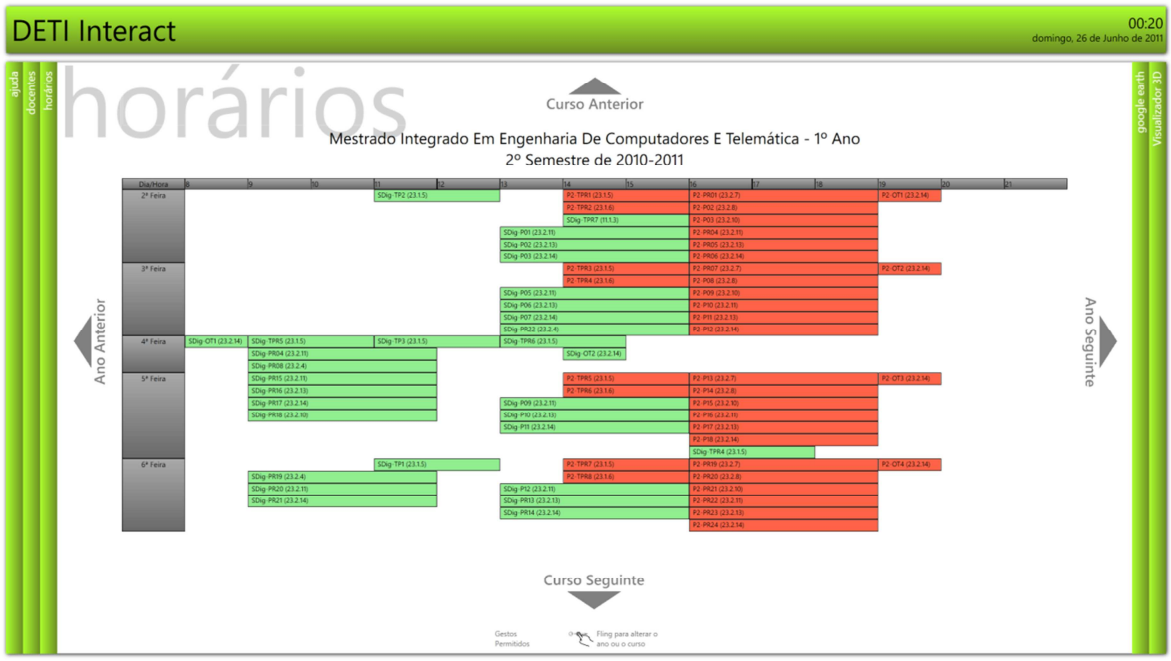

Figure 4.11 – The Timetables' page.

Analogous to the teacher control, an instance of the parser is created and every time the Changed event is triggered, the control draws to the screen. Data is passed to the control via a specific data structure containing subject name, classroom number, start time, duration, etc., called a `TimetableItem`. The timetable is represented by a Grid control where the rows represent days of the week, and the columns represent time. Before any work is done, the grid is configured for the appropriate degree program and year. The classes are then positioned on the grid, on their correct

places, colour-coded by course, so that they can be easily identified. In addition, since many classes can occur at the same, the grid's rows expand to accommodate them. The resulting timetable page can be seen in Figure 4.11.

4.3.3.2.1 Interaction

Interacting with the timetables via the Android device is limited to the Fling gesture. The user must swipe horizontally on the screen to navigate through the different years, or swipe vertically to change the degree program. Every time the user switches to a different timetable, the parser re-starts and process is repeated.

4.3.3.3 Google Earth Control

Another feature chosen to integrate the system is the inclusion of a Google Earth plugin in the application as it can also be used to validate the use of a mobile device for interaction with large displays, since it can offer a use for the touchscreen similar to what is already available on mobile devices and other touchscreen-equipped computers. Google Earth is presented on a page (Figure 4.12), and the user uses his/her phone to pan, tilt, or zoom.

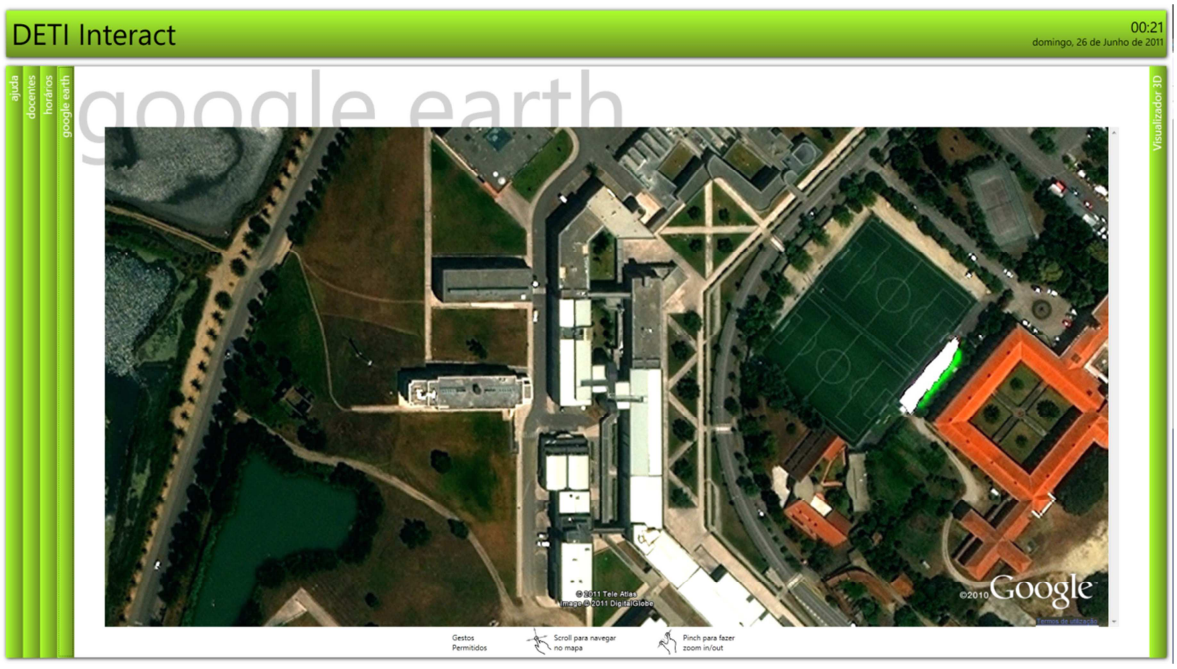

Figure 4.12 – The Google Earth page.

Google Earth works as a plugin integrated into a web browser. For developers, Google offers a Javascript API that allows web applications to host a Google Earth control. DETI Interact, however, is not a web application, meaning that there is no direct way to run the plugin in a native application.

In order to overcome this difficulty the developed control would have to be able to access the Javascript API from within the application. This was achieved in two steps. Firstly, it was necessary to find a way to draw a map on the screen, and allow interaction with it (panning, zooming, etc.). This was done via an HTML file. This file uses Google Earth's Javascript API to display the Google Earth control and works as a regular web page: it can be opened in a web browser, in which the Google Earth plugin is loaded and navigation within the plugin and interaction with it is as expected on any other page. The file also provides a small selection of Javascript functions that can be used to manipulate the map.

The second step consists in finding a way to render the HTML file inside the application, and using the provided functions to interact with the map. This is done by loading the file into a Web Browser control. This control renders the HTML content like any conventional web browser application, and using the provided `InvokeScript()` method, it is possible to call Javascript functions from within the HTML file. This resulted in the creation of new control to host the web browser

4.3.3.3.1 Interaction

Interaction with the map from the Android device is done as if the map had been drawn on the device's screen. Moving a finger in any direction will pan the map, and using the familiar pinch gesture, will zoom in or out, centred on the current position.

4.3.3.4 3D Viewer Control

The 3D Viewer control presents the content rendered by the XNA project mentioned in 4.3.2.2 within the WPF application. The 3D Viewer control is divided in two components: a panel to host the aforementioned D3DImage control and perform interoperation with XNA and DirectX, and a control to host this panel, which is tasked with animation and user interaction.

Viewer 3D Panel

The Viewer3DPanel hosts the D3DImage control and sets it up as the render target for the Model Viewer Library. This cannot be done directly although it is planned for future releases of both WPF and Silverlight (Microsoft 2011). This, however, can be achieved by using P/Invoke and Reflection to access specific DirectX and XNA content (Morril 2008).

In order to render a 3D scene into an image container inside the application, the render target of the XNA application must be switched to this container. This is done using the Interop Services

provided by the .NET platform. These services allow a developer to access features of native APIs, unavailable on the .NET framework, such as the Win32 API and DirectX. In the Viewer3D panel, P/Invoke is used to fetch the `IDirect3DTexture9` and IDirectDevice9 interfaces from the DirectX API. In addition, Reflection is used to get the private `ChangeDevice()` and `Initialize()` methods from the XNA API.

The Viewer3D Panel control can now access the `ModelViewer` library discussed in 4.3.2.2, and set the render target to the image container it hosts.

Viewer 3D

The Viewer3D is a control that wraps around the previously created panel. It was developed using the MVVM pattern. Its View Model is databound to the mentioned XNA application, while the view is responsible for animation and user input handling.

The Viewer3D control displays a list of all the three-dimensional models available in the system. It supports a selection mode similar to the one developed for the Teacher page. Once one of the models is selected, the Viewer3D Panel is displayed and interaction with the model is possible (Figure 4.13).

Figure 4.13 – The 3D Viewer Control, displaying a model synchronized with the mobile device's digital compass.

4.3.3.4.1 Interaction

Interaction with the Viewer page is similar to the interaction developed for the Teacher page. This page displays a list of the available models, in which the user can scroll of fling to browse its contents. A Long Press enters and exits a selection mode that allows the user to load a desired model via a Tap on the device's touchscreen. When a model is loaded, the Viewer 3D Panel is displayed, and the user can use his phone to manipulate the model. Orientation changes on the phone will directly affect the model, as if the user was holding the displayed model in his/her hands.

4.4 Deployment

In order to get the system working, the desktop application had to be installed and the mobile application had to be installed on the phone. The installation of the desktop application faced a series of problems. Since the application heavily depends on external sources (the Web Services from the DSD platform) and some features from the .Net framework, it was not possible to simply run the binary on the target machine. The application had to be installed in order for all the dependencies to be resolved. This was done using the provided ClickOnce functionality. This feature generates an executable binary for the application along with the needed assemblies and a configuration file. When the binary is run, the application is automatically installed and run.

Once the application was installed, it was also necessary to configure the machine so that it would run unattended. This involved setting up an automatic logon, configuring the anti-virus software and configuring the computer's software and services that run automatically during boot, so that DETI Interact can run without interruptions.

4.5 Conclusion

In Chapter 3, the system architecture was detailed, along with the two applications that would make up the system. The mobile device would work simply as a peripheral; therefore, the developed mobile application simply gathers data from the sensors and sends it via a Bluetooth connection. The desktop application then processes this data and interprets the gestures. The main function of the desktop application is to present content to the user. It connects to the required services and populates the screen. The system was developed and structured in a way that simplifies future modifications and extensions, so that additional content and interaction methods can be introduced to the system.

5 User Evaluation

In this chapter, the evaluation of the system will be discussed. In addition to user testing of the system, an application to evaluate Fitts' Law was developed to compare the use of a mouse versus the use of a mobile device as a means of interaction, in a series of selection tasks.

5.1 User Testing

An important stage in the development of an interactive system lies in user testing. Since the users will be the primary clients of the system, it must be ensured that they understand how to operate the system and enjoy using it. These tests assist in identifying errors in the system and problems in the user interface, which can then be addressed in order to deploy a fully functional version of the system.

DETI Interact proposes the use of a smartphone to control the display of information on a large screen. This was also studied during the development of DetiGuide (Palha 2010), the results, however, were inconclusive, consequently, with DETI Interact instead of proceeding straight to test the developed system, it was decided to conduct a preliminary testing session to evaluate Fitts' Law. The model developed by Paul Fitts relates the size of a target and its distance to the mouse cursor with the time it takes to select it. This would enable testing different usage scenarios where a smartphone could be used to control a computer. In addition to this test, a conventional test to evaluate the usability of the system was also organized.

5.1.1 Fitts' Law

In order to compare the ease of use of any given system using a mouse or a mobile device, an application was developed to evaluate Fitts' Law. Fitts proposed a formal relationship that models speed-accuracy trade-offs in rapid, aimed movement. This model predicts that the time required for a user to move to a target area is related to the distance and size of the target. It has been formulated mathematically in several different ways; the following equation, known as the Shannon formulation is preferred since it always provides a positive rating for the index of difficulty for each task (MacKenzie and Buxton 1992).

$$MT = a + b \log_2\left(\frac{2A}{W} + 1\right)$$

In the equation, **MT** represents the movement time to hit the given target, with **A** and **W** representing the distance to the target (amplitude), and the target's width, respectively. Constants **a**, and **b** are determined empirically with user testing. Once the results are processed, **a** expresses the start/stop time of the devices, and **b** represents the inherent speed of the device. The term $\mathbf{\log_2\left(\frac{2A}{W} + 1\right)}$ represents the index of difficulty, **ID**. In addition, $\mathbf{1/b}$ is called the index of performance, **IP**, of a device.

To compare the performance between the use of a mouse and mobile device, an additional application was developed, using .NET and WPF. This allowed for the reuse of components such as the Control library (described in chapter 4.3.2.3), which gives access to the Bluetooth stack as well as the gesture handlers. This application generates circular objects on de screen, which the user must click. The size and position of each circle on the screen is random, in order to randomize the index of difficulty. To ensure a wide range of data, a total of 50 targets are generated. There were five different circle sizes used on the application: 20px, 40px, 80pc, 120px, and 240px in diameter. The smaller sizes, 20px and 40px, were displayed with roughly 10mm and 20mm, which are confortable sizes to be used for finger-based interaction on a touchscreen (Microsoft 2011).

There were three different interaction methods available:

- The mouse, to be used as a reference for the other tests – the mouse is used to move the cursor, and the mouse buttons used to click on the targets;

- The touchscreen on the mobile device – the touchscreen is used to move the mouse cursor as a computer touchpad would, and tapping the touchscreen would send a click action to the application;

- The digital compass on the mobile device – device orientation manipulates the mouse cursor like a joystick, and tapping the screen clicks the targets.

Given the random nature of the tests, each user can perform the three tests. Since both mobile interaction methods are slower than the computer mouse, the mouse's sensitivity was reduced, to approximate the speed at which the others operate. All users were positioned at the same distance from the display, delimited by a table that would provide a surface for the mouse. Users remained standing throughout the tests.

5.1.1.1 Results

The tests were performed with 34 computer engineering students, resulting in 1700 results per test. All the tests were completed successfully, with the clear worst interaction being the use of the digital compass. The main problems with both methods of interaction on the mobile device were directly related to the accuracy in the collected values: the touchscreen has a much lower resolution than the computer mouse, and the digital compass, even in its fastest polling state, was too slow to manipulate the cursor comfortably.

In general, both methods have an overall worse performance when compared with the computer mouse. During the test, it was observed that when tapping the touchscreen users would often slightly move the cursor. This affected both interaction methods. The users felt that the smaller targets (roughly 1cm in diameter) were the hardest to click. In this case, both methods were distressing since very slight adjustments on the touchscreen were often detected as a Long Press by the Android OS, and the sensibility of the compass would have the users circling around each target without being able to position the cursor correctly. However, using the compass on these smaller targets still offered better precision. Apart from being more efficient with the touchscreen, some users enjoyed using the compass, since they could tap on the screen while tilting the device. In the end however, the preferred method of interaction was the touchscreen.

The results of this series of tests can be seen plotted in Figure 5.1. This graph shows the time it took for each user to select each target, using the three interaction methods. The horizontal axis expresses the index of difficulty, which relates the size of a target with the distance from the cursor.

The mouse interaction, used as a baseline, offers the best performance as it can be seen in the chart. Although the surface on which the mouse was placed was not optimal, which can be seen in some of the values, users were able to click the targets quickly. The interaction methods using the digital compass or the touchscreen on the mobile device show a worse performance, with the digital compass displaying the worst results.

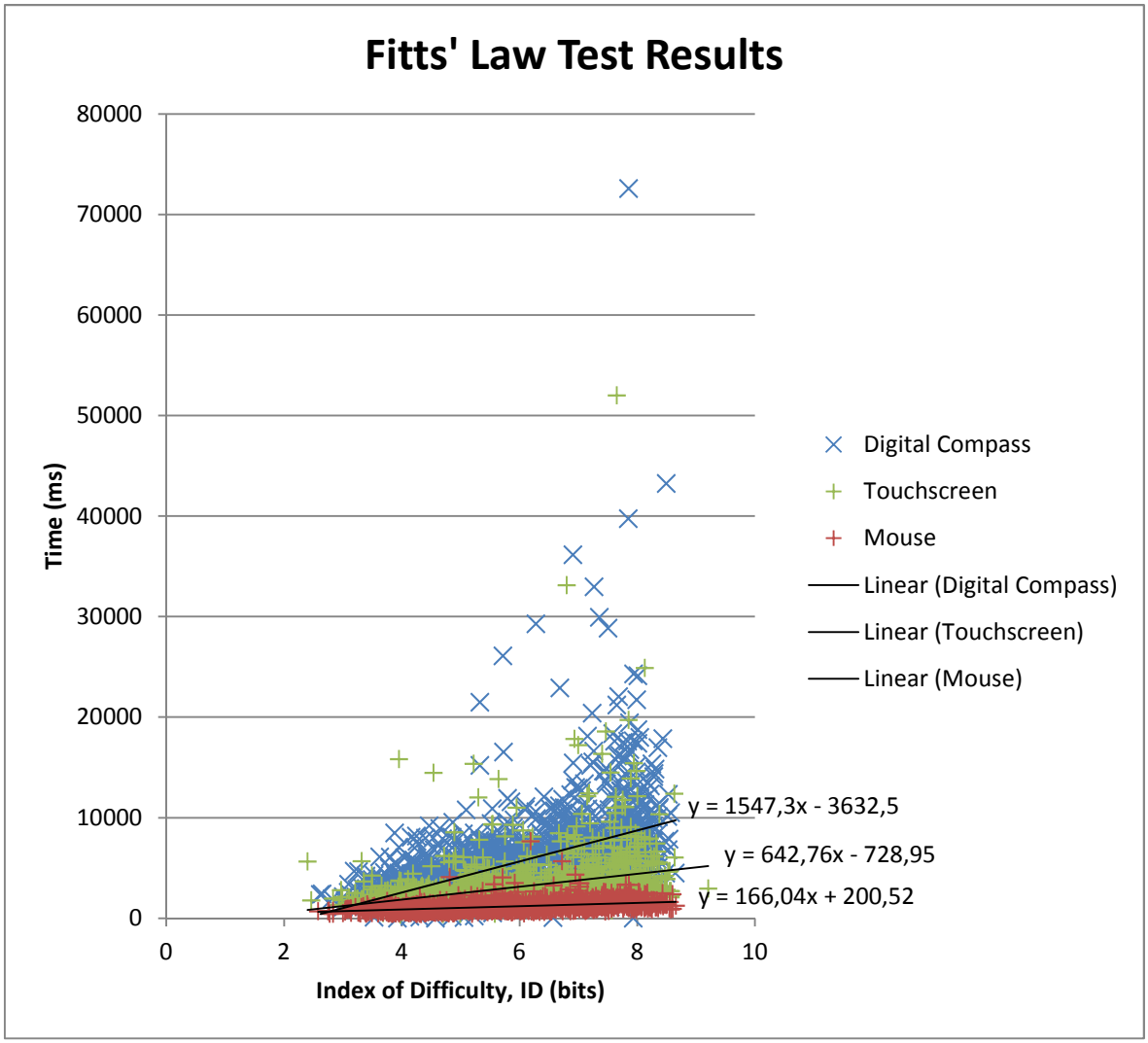

Figure 5.1 – Plot of the results obtained in the Fitts' Law test, along with a regression line for each method.

In addition to evaluating the time it took for the users to click on a target of a certain difficulty, it was also interesting to relate the elapsed time with the size of the targets. Figure 5.2 shows how long it took the users to click on each target, given its size. As specified previously, the size of the targets ranged from 20px to 240px. The graph shows that, as anticipated, users spent more time in attempts to click the smaller targets, with the larger targets earning the better results. This shows

that when creating user interfaces for this type of interaction, buttons, links, or other controls should have a considerable size, in order to provide the user with a better experience.

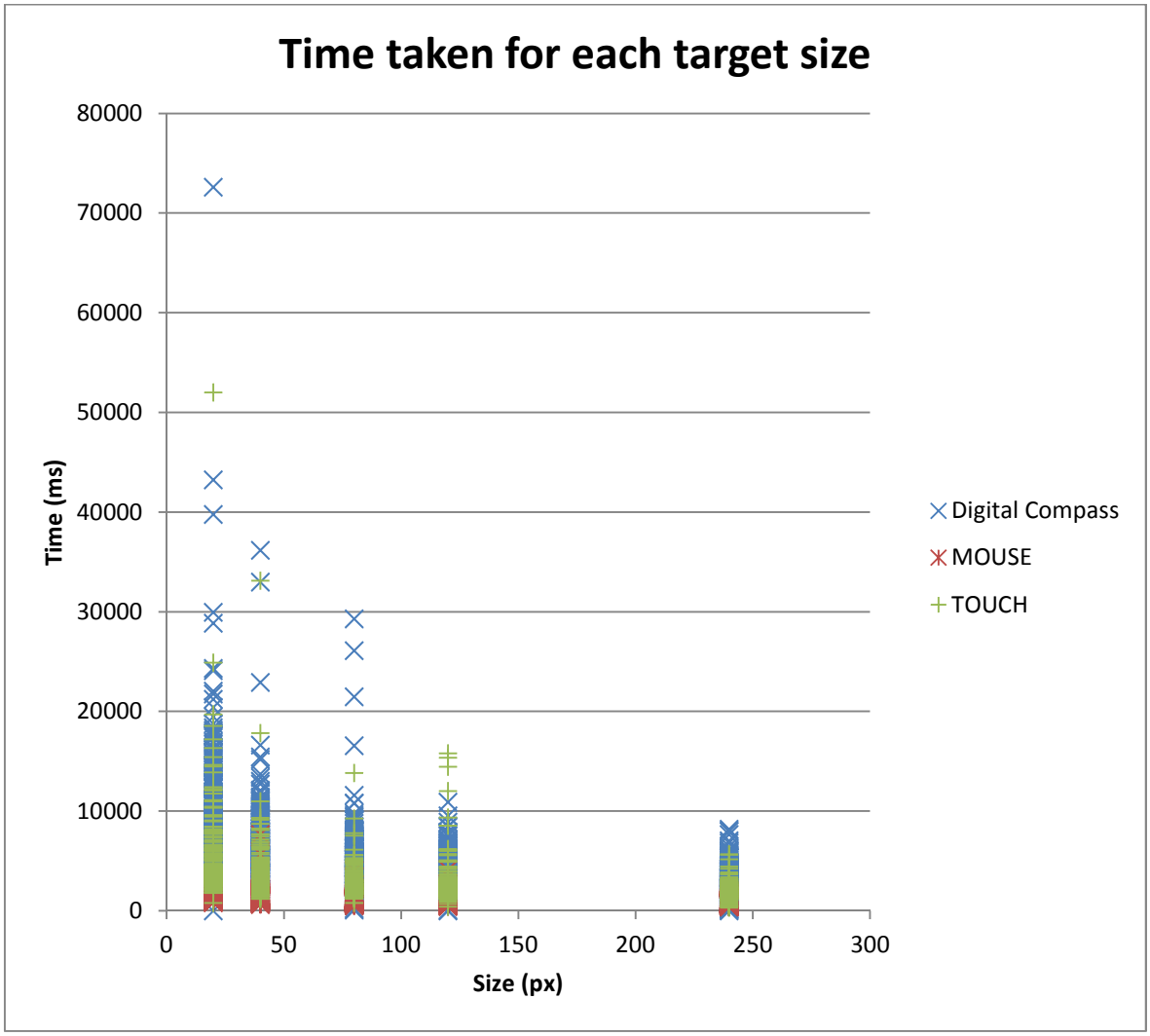

Figure 5.2 – Plot that relates the time taken to select each target with its size.

5.1.2 DETI Interact

The tests based on Fitts' Law assist in evaluating the touchscreen and digital compass of a mobile device as input methods, and validating their use in different scenarios. Nevertheless, interaction with the DETI Interact system must also be tested. Since many users are not familiar with modern smartphones, and given that touchscreen computers are more common in the scenario explored by this system, DETI Interact was tested on a computer with a touchscreen, as well as the large screens in the department's atrium. The touchscreen computer was not used during the Fitts' Law tests for two reasons: it had a much smaller resolution, which would affect the index of difficulty for each

target, and it allowed users to immediately tap the circle, which would evaluate his/her response time instead of the device's performance.

Interaction with the large screens was done with the smartphone, with the gestures discussed in chapter 4. Its touchscreen is used to control the content presented in each page, while the digital compass is used to alternate between the available pages.

In order to test the system, a list of tasks was prepared for the users to carry out. These tasks consisted of operations of varying difficulty, and ensured that the users would use all the provided features, simulating realistic usage of the system. Users were asked to complete these tasks and rate them regarding their difficulty. Once the tests were over, the users were asked to fill a small questionnaire where they could rate their experience and provide some comments about the system. The proposed tasks were the following:

1. Connect to the system;
2. Find a phone extension for a certain teacher;
3. Find more detailed data for another teacher by accessing his or her website;
4. Check the classroom for a given course;
5. Navigate to a location on Google Earth;
6. Manipulate a three-dimensional model.

The tasks were the same on both computers; however, the Google Earth plugin was not operating correctly on the touchscreen computer, which resulted in the termination of the application. This led to the creation of two different versions of the application, a fully featured version, running on the large screen, and a version without the Google Earth plugin, running on the touchscreen computer. Users on the touchscreen were asked to skip the task referring to navigation using Google Earth.

5.1.2.1 Results

This test was performed with 22 computers engineering students. All the tasks were completed successfully, with both devices showing favourable results. Figure 5.3 shows the average results for the difficulty of each task, with one being very hard, and five being very easy. Apart from the tasks

that the touchscreen could not run – connecting via the mobile application and using Google Earth – it can be seen that the scores between both computers are very similar.

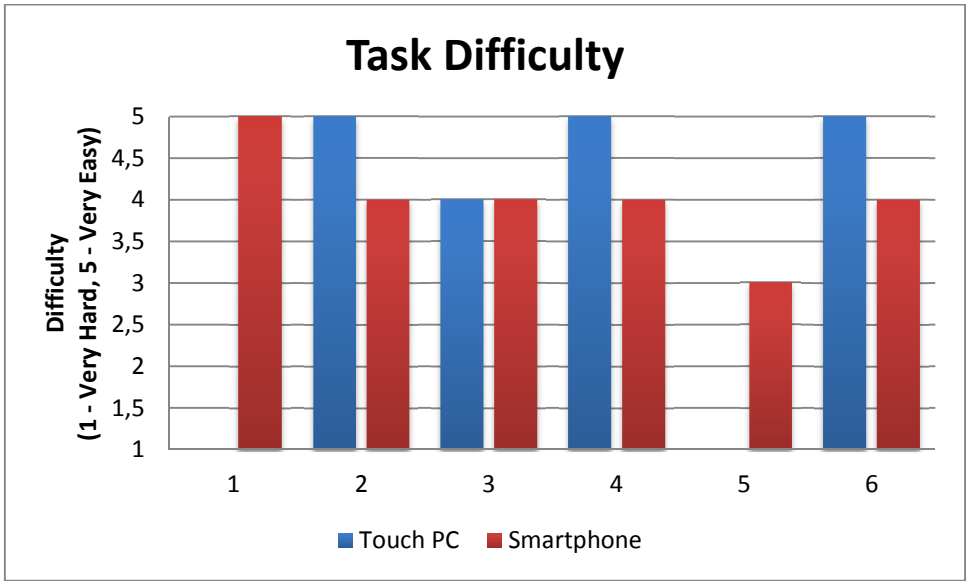

Figure 5.3 – Difficulty of the tasks, expressed by the users, comparing the two interaction methods.

When users connect to the system, a help page is automatically displayed and, in addition, assistance on how to interact with each page is provided at the bottom of the pages. These measures ensure that the users do not feel lost or confused when operating the system. During the observation of the tests, the appropriate gestures were performed. One of the main complications with the mobile device was related with the direction of movement. Since not many users were familiar with modern interaction methods, such as scrolling a list of items on a mobile device, users sometimes expected the system to respond differently. For instance, scrolling down the list of teachers is done by moving the finger upwards on the touchscreen. This simulates the action of dragging the list upwards, in order to reveal the content below. This type of gestures is currently used on most touch-based mobile devices, which led to them being implemented this way on the system. Additionally, users also made mistakes when alternating between pages, which is done by tilting the device sideways. They would either tilt the device in the wrong direction, or attempt to change pages with a horizontal swipe on the touchscreen. Another complication common with most users was interacting with Google Earth. As explained in section 4.2, the pinch gesture was unavailable in the used version of the API, which led to this feature being implemented from scratch. The resulting gesture was very sensitive which left users sometimes feeling out of control.

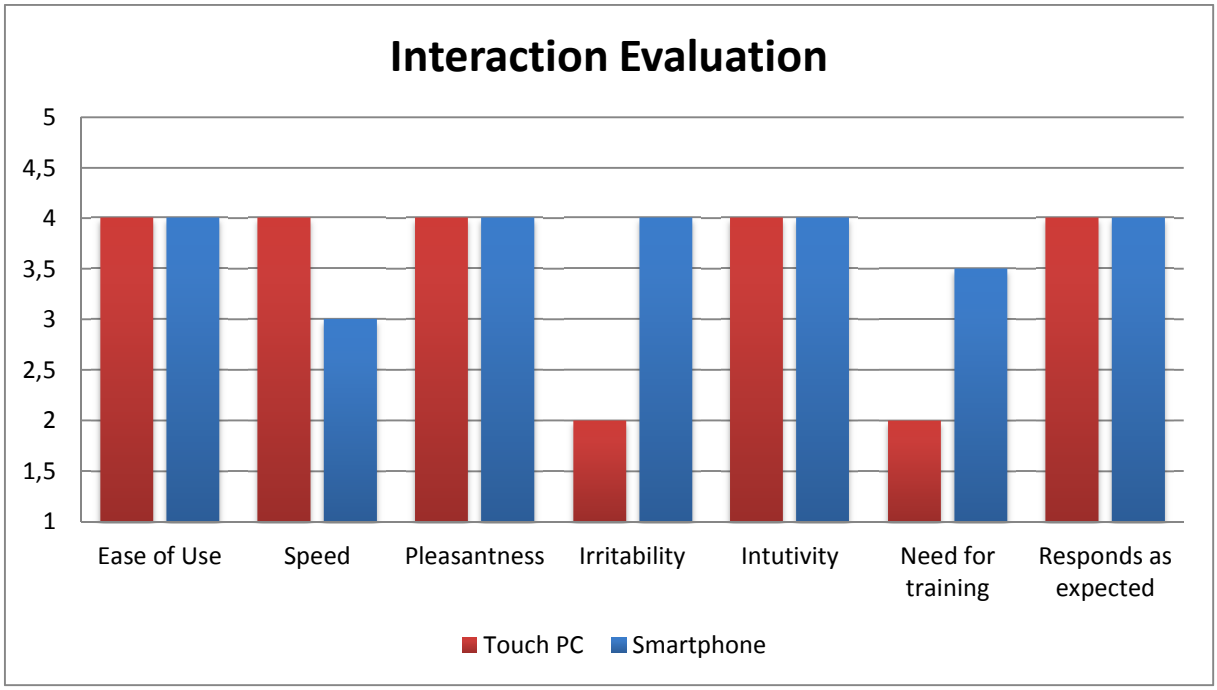

Figure 5.4 - Results of the queries answered by the users after the completion of the tasks.

Following the testing session, users were asked to fill a small survey regarding their experience when using the system. In this survey, they would rate the system concerning ease of use, speed, pleasantness, irritability, intuitiveness, need for training, and responsiveness. The results indicate that the overall experience using the smartphone was inferior when compared to the touchscreen (Figure 5.4). While most users felt that the system was intuitive and easy to use, the lowest scores lie in the speed, irritability and need for training. This, however, was not unexpected. The overall speed of the system was affected by the amount of data being drawn to the screen. As explained on section 3.4.2, the computer running the software presented low performance. With the list of teachers consuming a lot of memory, and the 3D Model Viewer's use of the GPU, which was aggravated by the use of reflection in the code, the system suffered severe hits on performance. In addition, as explained above, some users tried tilting the device, or scrolling in the wrong direction, and most of the users where overwhelmed by the sensitivity of the pinch gesture in the Google Maps page.

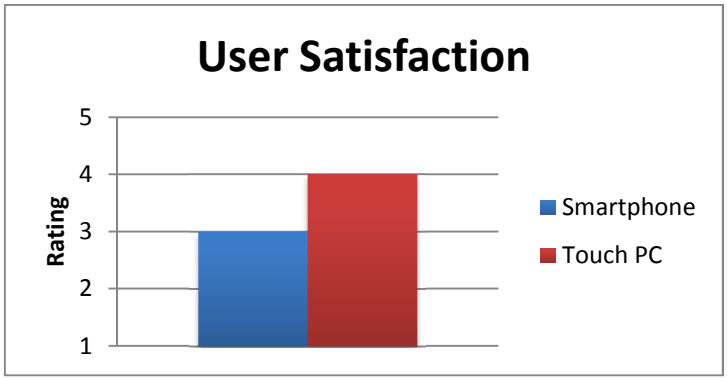

Figure 5.5 – Satisfaction of the users when using either interaction method.

Users were also asked to rate their overall satisfaction with the system (Figure 5.5). Once again, the use of the smartphone received worse ratings. When asked for their preference, users also selected the touchscreen-equipped computer as their favourite, although some users did choose the smartphone over the touch PC (Figure 5.6).

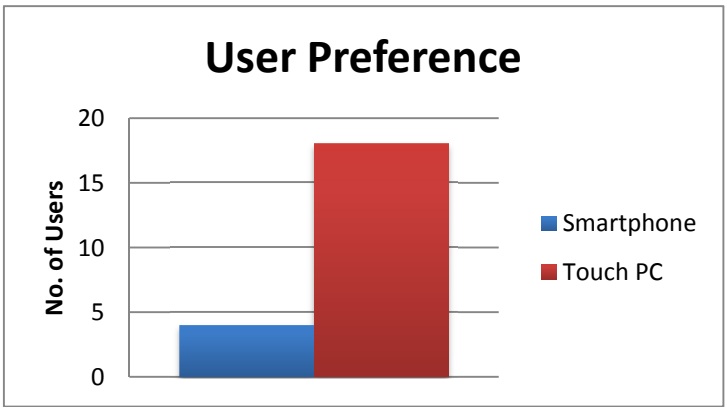

Figure 5.6 – User preference on the interaction method.

In the survey, users had the opportunity to write some comments and give suggestions on how to improve the system. Almost all the comments criticized the sensitivity of the zoom gesture on Google Maps. Some suggested that a double tap gesture could be added to the system to complement the zooming gesture. There were also some references to the titles of the pages, displayed vertically on the sides of the pages, had a very small font. When asked to suggest new features, nearly all the participants requested that the menus for the canteens be displayed on the application. In addition, some news relative to the department, as well as the university, could also be displayed.

5.2 Discussion

Testing the system with users is an important stage when developing software. The objective is to identify errors in the system and determine areas where improvements can be made. For this project, two different testing waves were organized. The first one addressed Fitts' law and its implications. The second involved testing the developed system. Testing was always done with different interaction methods, in order to compare the system with other real world scenarios.

As explained previously, Fitts' Law establishes a relation between the size of a target and its distance from the cursor with the time it takes a user to select it. Data gathered from this test can be used to design better user interfaces. Since DETI Interact proposes the use of a smartphone to control information displayed on a large screen, this test compared the use of the touchscreen and the digital compass with the use of the computer mouse. Both of the studied methods are naturally more arduous when compared with the mouse, but between them, the use of the digital compass offers the worst control method. From the results, it can be concluded that the digital compass should not be used for tasks that require some dexterity. The touchscreen, on the other hand, fares well in these situations.

Considering these results, the interaction methods chosen for DETI Interact – using the touchscreen for most of the operations, and the digital compass to alternate between pages – where considered valid, although some adjustments or alternative gestures could also be provided. In addition, the results also indicate that in scenarios where the user might need to control a cursor or select certain objects, these must have a practical size, with at least 100 pixels in diameter. Furthermore, alternative methods of interaction might be considered. These methods might include different speed or acceleration for the cursor movement, depending on the distance to the target, or using an approach where the targets generate gravity, which would assist the user in the selection.

After processing all the data from the first wave of tests, users were asked to participate in a second wave, in order to evaluate the usability of DETI Interact. While this second wave of tests did not provide the most favourable results towards the system, these were close to those obtained from similar devices (the touchscreen-equipped computer). Considering the results gathered from the previous testing session, concerning Fitts' Law, and the observed reactions of the users, the chosen library of gestures was deemed fitting. While the learning curve for the system may be steeper when compared with using direct interaction methods, users still felt that the system was easy to use, albeit with some initial guidance.

Some users complained about the slow performance of the system. As explained above, this had to do with the implemented 3D Model Viewer, alongside the low performance of the system. This feature was used as a means of comparing three-dimensional interaction techniques using a mobile device equipped with a digital compass, accelerometer, or gyroscope, with direct methods of interaction, such as the touchscreen computer. Given its utility for the department, and the limited availability of 3D models, this feature was removed from the version of the system currently running in the computers of the department. The amount of data displayed on the teachers' page also hinders system performance, although the hit is less severe. Another complaint was concerning the sensitivity of the pinch to zoom gesture on the mobile device. The ideal solution for this problem would be to use the API for Android 2.2, which includes the appropriate functions.

Since the testing, some changes were made to the application, mostly to the user interface such as, some font sizes were increased to improve readability, and the titles of the pages were added to the background of the pages to better situate the users. Visual help was also improved.

6 Discussion

6.1 Conclusion

The main purpose behind DETI Interact was to develop a system that displayed information concerning the Department of Electronics, Telecommunications, and Informatics of the University of Aveiro, and allowed users to interact with this information via an Android smartphone. This would allow users to interact with a system that did not support any type of direct or indirect interaction, such as touchscreens. The system consisted of an application that ran on a computer connected to a large display, which is located in the lobby of the department. An application for Android based devices was also developed which was responsible for the communication between the mobile device and the computer.

6.1.1 Development

As described in chapters 3 and 4, the system communicates with the DSD platform, the department's server that holds information regarding lecturers, students, classes, etc., to gather data relating to the department. It also uses the Google Earth API to display a map plugin and the XNA framework to display and allow interaction with three-dimensional objects.

The desktop application was developed using .NET and WPF. These technologies were helpful in prototyping and developing the system rapidly, as well as offering a greater performance. Given the low hardware specifications of the target machines, it might have been worthy to attempt to develop the system in a lower level language (using C++ with QT for the user interface, for example). An

attempt was made using Windows Forms, but the large amount of data drawn on screen was significantly worse, when compared to WPF. Using WPF was also helpful in developing a system that was easily modifiable and extensible, for its use of the MVVM pattern.

The Android mobile application was developed in Java. While Android offers a good development environment when compared to other mobile platforms, its programming model, using the MVC Pattern, and the provided API, feels cluttered, which degraded the developing experience. The main problems were encountered with the Bluetooth communication. It was chosen for its simplicity and lack of configuration needed by the user, but the API provided in the Android SDK damaged the experience that could be provided to the users of DETI Interact.

Overall, the development of DETI Interact did not face any major setbacks, due to well-defined system requirements and suitable choice on the development platforms.

6.1.2 Results

The system was deployed to the lobby where it has been running since January 2011. During this time, it has been receiving updates and generating usage logs, which assisted in its development and maintenance.

The system was later tested with users to evaluate the performance of the smartphone as an input peripheral, as well as the usability of the application. The first testing wave targeted the evaluation of the performance of the smartphone. It was based upon Fitts' Law and yielded interesting results, which are described in detail in Chapter 5. The results suggested that the use of the touchscreen of a mobile device could be an appropriate input peripheral, while the use of the digital compass could be more problematic. This corroborated the choices made during development, where the compass was used for switching pages, by tilting the device sideways, and the touchscreen was used to interact with the content presented on each page.

A second wave of testing evaluated the usability of DETI Interact. Users were given a series of tasks to fulfil and then rate their difficulty. In the end, they also participated on a small survey where they rated their experience and commented the system. The results were compared with the use of a touchscreen-equipped computer, which would be a more familiar device. While the touchscreen computer received better results, these remained close to the results obtained from the smartphone. Users thought that, while the system was easy to use, some usage scenarios were

more irritating in the smartphone, and that the learning curve was steeper. While at first glance the results seem unfavourable towards the smartphone, it must be taken into account that these were compared with other already well-established methods of interaction, such as the computer mouse, with which users are more experienced. However, the mouse itself took thirty years to become a ubiquitous device (Buxton 2007). Natural User Interfaces (NUI), such as those explored by voice, gesture, and touch, albeit quickly becoming embraced by the industry and the public, are still struggling to become ubiquitous. They are still criticized for not being as good as a computer mouse for most tasks, even though they might be superior in other tasks. As William Buxton says: "everything is best for something and worst for something else", also adding, "the trick is knowing what is what, for what, when, for whom, where, and most importantly, why. Those who try the replace the mouse play a fool's game. The mouse is great for many things. Just not everything" (Buxton 2007). It was not unexpected that the interaction performance with the mobile device would be worse, but it allows for a good method of interaction in a situation where other alternatives would not be viable or would require a greater investment.

In conclusion, DETI Interact proposed to study approaches to interaction with large displays, and implement and information system on the lobby of the Department of Electronics, Telecommunications, and Informatics of the University of Aveiro. Although the results could have been superior, the system received good reviews, and is currently running on a on a computer in the department. While direct manipulation of data on a remote screen via a smartphone might appear to be an arduous task for users, given the impossibility of using a computer mouse, an appropriate user interface and set of gestures can make a system usable, with a promising user experience.

6.2 Future Work

As with any software project, there are always areas that can be improved, and features that can be added. The results and comments gathered from the testing sessions were invaluable in finding areas in need of improvement. The principal complaint about the system was in regards to the zoom gesture in the Android device. This had to do with the gesture not being available in the API and having to be written from scratch. While the sensibility can be improved, version 2.2 of the API includes the classes that handle this gesture, which would provide the best user experience.

In addition, alternative gestures can be implemented, such as the Double Tap, also a common in zooming tasks. Furthermore, the system could use and extended gesture library. For instance, some

users would try to use the Fling gesture to alternate between pages, instead of tilting the device. Since the touchscreen allows detection of multiple points, a two-finger swipe can be implemented.

In respect to new features, users suggested that the system could also display the menus for the canteens, ticket information from the university's administration services, as well as news from the department. Additional information such as exam dates, vacant classrooms, integration with the university's library, was also suggested by other students.

Apart from the functionalities suggested by the users, other features were already being considered. One interesting feature would be to make the system context-aware. Once a user connected to the system, he would be identified and specific information could be displayed, such as his timetable, exam dates, project deliveries, etc. This would require further integration with the university's systems such as PACO (*Portal Académico Online*) and Moodle. Additional information could also be displayed on the mobile device's screen. The mobile device is currently used solely as an input peripheral, but the available screen can also be used for information display. As stated in chapter 4, Android 2.3 allows for the use of Insecure Bluetooth Sockets. This removes the need for the devices to pair, which would improve the user experience on the first connection to the system. However, using a more recent version of Android (whether it is 2.2, for the scale gesture, or 2.3, for the insecure Bluetooth sockets) will exclude users running older versions.

In addition, other mobile platforms could also be included. While not all provide access to the Bluetooth stack, future updates could bring such functionality, and other communication methods are available (e.g. WiFi, Sockets). Furthermore, given the results obtained from the first testing session, other interaction methods could be added to the system, such as the ability to control the mouse cursor. This would allow users to navigate webpages, for example, as opposed to simply viewing them as is currently done. It could also improve selection-based tasks, such as those existent in the teacher page. DETI interact was developed in a way that interaction with the system is not dependent on any device, which allows for the system to support additional devices such as video cameras or the Microsoft Kinect device. The system could also be extended in order to allow multiple users in a collaborative environment, as well as the possibility of spanning across additional displays. Augmented reality is also an option that can enrich the experience of the user. Testing also revealed that the mobile device offers a suitable method of interaction for 3D content, meaning that further applications of three-dimensional content should be studied.

References

Anthony, S. (2010). "Microsoft Kinect controlling Windows 7, exciting proof of concept." Retrieved 25/11/2010, from http://downloadsquad.switched.com/2010/11/22/microsoft-kinect-controlling-windows-7-exciting-proof-of-concep/.

Boring, S., D. Baur, et al. (2009). "Touch Projector: Mobile Interaction through video." Proceedings of CHI 2010, Atlanta, GA, April 10-15: 2287-2296.

Buxton, W. (2007, 21/03/2011). "Multi-Touch Systems that I Have Known and Loved." Retrieved 18/06/2011, from http://www.billbuxton.com/multitouchOverview.html.

Campos, D. (2008). Distribuição de Serviço Docente: back e front office Web. Departamento de Electrónica, Telecomunicações e Informática, Universidade de Aveiro. **MSc:** 148.

CellPhonesMarket. (2011). "Google's new SDK features an API for "insecure Bluetooth socket connections"." Retrieved 21/05/2011, from http://www.cellphonesmarket.com/news/googles-sdk-features-api-insecure-bluetooth-socket-connections/.

Cheng, K. and K. Pulo (2003). Direct interaction with large-scale display systems using infrared laser tracking devices. Proceedings of the Asia-Pacific symposium on Information visualisation - Volume 24. Adelaide, Australia, Australian Computer Society, Inc.**:** 67-74.

Cull, T. (2010). "Researchers Highlight Recent Uptick in Java Security Exploits." Retrieved 14/02/2011, from http://www.infoq.com/news/2010/10/java-exploit-uptick.

DailyTech. (2011). "Android Overtakes Symbian as World's No. 1 Smartphone Platform." Retrieved 14/02/2011, from http://www.dailytech.com/Android+Overtakes+Symbian+as+Worlds+No+1+Smartphone+Platform/article20787.htm.

Finke, M., A. Tang, et al. (2008). Lessons learned: game design for large public displays. Proceedings of the 3rd international conference on Digital Interactive Media in Entertainment and Arts. Athens, Greece, ACM**:** 26-33.

Fling, B. (2009). Mobile Design and Development, O'Reilly.

Foot, P. "In The Hand .NET components for mobility." Retrieved 04/11/2010, from http://inthehand.com/.

GIZMODO. "Orange Shows Off Gesture Based Interaction Screen, Touch Screens Look On in Horror." Retrieved 04/11/2010, from http://gizmodo.com/346845/orange-shows-off-gesture-based-interaction-screen-touch-screens-look-on-in-horror.

Google. (2005). "Google Earth." Retrieved 13/06/2011, from http://www.google.com/earth/index.html.

Google. (2010). "Features - Google TV." Retrieved 24/11/2010, from http://www.google.com/tv/.

Google. (2011). "What is Android? | Android Developers." Retrieved 12/04/2011, from http://developer.android.com/guide/basics/what-is-android.html.

Hardy, R. and E. Rukzio (2008). Touch \& interact: touch-based interaction of mobile phones with displays. Proceedings of the 10th international conference on Human computer interaction with mobile devices and services. Amsterdam, The Netherlands, ACM**:** 245-254.

Hardy, R. and E. Rukzio (2008). Touch \& Interact: touch-based interaction with a tourist application. Proceedings of the 10th international conference on Human computer interaction with mobile devices and services. Amsterdam, The Netherlands, ACM**:** 531-534.

Ideum. (2011). "Supported Multitouch Gestures." Retrieved 16/06/2011, from http://gestureworks.com/support/supported-gestures/.

Kaviani, N., M. Finke, et al. (2009). "Encouraging Crowd Interaction with Large Displays using Handheld Devices " Proceedings of CHI 2009, Boston, MA, April 4-19.

MacKenzie, I. S. and W. Buxton (1992). Extending Fitts' law to two-dimensional tasks. Proceedings of the SIGCHI conference on Human factors in computing systems. Monterey, California, United States, ACM**:** 219-226.

Madhavapeddy, A., D. Scott, et al. (2004). "Using Camera-Phones to Enhance Human-Computer Interaction." Proceedings of Ubiquitous Computing (Adjunct Proceedings: Demos): 1-2.

Microsoft. (2009). "Microsoft .NET Framework." Retrieved 13/06/2011, from http://www.microsoft.com/net/.

Microsoft. (2010). "Introduction to WPF." Retrieved 20/08/2010, from http://msdn.microsoft.com/en-us/library/aa970268.aspx.

Microsoft. (2010). "Kinect - Xbox.com." Retrieved 25/11/2010, from http://www.xbox.com/en-US/kinect.

Microsoft. (2011). "Kinect for Windows SDK from Microsoft Research." Retrieved 16/06/2011, from http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/.

Microsoft. (2011). "Microsoft Expression Blend® 4 | Silverlight | Rich Internet Applications | XAML | WPF Applications | .NET Platform | TFS | VB | C# | Microsoft® Expression®." Retrieved 13/04/2011, from http://www.microsoft.com/expression/products/Blend_WhatIsExpressionBlend.aspx.

Microsoft. (2011). "Silverlight 5 beta: The Official Microsoft Silverlight Site." Retrieved 13/06/2011, from http://www.silverlight.net/getstarted/silverlight-5-beta/.

Microsoft. (2011). "Touch." Retrieved 25/05/2011, from http://msdn.microsoft.com/en-us/library/cc872774.aspx.

Mihailescu, D. (2010). "Benchmark start-up and system performance for .Net, Mono, Java, C++ and their respective UI." Retrieved 14/02/2011, from http://www.codeproject.com/KB/dotnet/RuntimePerformance.aspx.

Miyaoku, K., S. Higashino, et al. (2004). C-blink: a hue-difference-based light signal marker for large screen interaction via any mobile terminal. Proceedings of the 17th annual ACM symposium on User interface software and technology. Santa Fe, NM, USA, ACM**:** 147-156.

Morril, J. (2008). "XNA, Meet WPF." Retrieved 04/11/2010, from http://jmorrill.hjtcentral.com/Home/tabid/428/EntryId/259/XNA-Meet-WPF.aspx.

Palha, R. (2010). Interacção entre um dispositivo móvel e um ecrã de grandes dimensões. Departamento de Electrónica, Telecomunicações e Informática, Universidade de Aveiro. **MSc**.

PhoneGap. (2008). "PhoneGap." Retrieved 11/04/2011, from http://www.phonegap.com/.

Richter, J. (2010). CLR via C#, Microsoft Press.

Samsung. (2010). "Samsung Internet @ TV." Retrieved 24/11/2010, from http://www.samsung.com/uk/internet-tv/.

Seewoonauth, K., E. Rukzio, et al. (2009). Touch & connect and touch & select: interacting with a computer by touching it with a mobile phone. Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services. Bonn, Germany, ACM**:** 1-9.

Stogaitis, M. and M. Sun. (2008). "Gmote - Android Remote." Retrieved 21/11/2010, from http://www.gmote.org/.

Tang, A., M. Finke, et al. (2008). Designing for bystanders: reflections on building a public digital forum. Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems. Florence, Italy, ACM**:** 879-882.

TheAlternative. "The Alternative Homepage." Retrieved 21/11/2010, from http://www.thealternative.co.uk/.

Thurrot, P. (2010). "More work on Chapter 4 and notes about the origins of Metro." Retrieved 14/05/2011, from http://windowsphonesecrets.com/2010/04/04/more-work-on-chapter-4-and-notes-about-the-origins-of-metro/.

Thurrott, P. (2010). "Xbox 360 Dashboard Screenshots Gallery 1." Retrieved 14/05/2011, from http://www.winsupersite.com/article/product-review/xbox-360-dashboard-screenshot-gallery-1.

Totilo, S. (2010). "Natal Regonizes 31 Body Parts, Uses Tenth of Xbox 360 "Computing Resources"." Retrieved 01/03/2011, from http://kotaku.com/5442775/natal-recognizes-31-body-parts-uses-tenth-of-xbox-360-computing-resources.

Underkoffler, J. (2010). "John Underkoffler points to the future of UI." Retrieved 01/03/11, from http://www.ted.com/talks/john_underkoffler_drive_3d_data_with_a_gesture.html.

Vogel, D. and R. Balakrishnan (2004). Interactive public ambient displays: transitioning from implicit to explicit, public to personal, interaction with multiple users. Proceedings of the 17th annual ACM symposium on User interface software and technology. Santa Fe, NM, USA, ACM**:** 137-146.

W3C. (2004). "Web Service Glossary." Retrieved 04/11/2010, from http://www.w3.org/TR/ws-gloss/.

Wired. (2009). "5 Nifty Apps That Turn Your Android Into A Universal Remote." Retrieved 21/05/2011, from http://www.wired.com/gadgetlab/2009/11/5-nifty-remote-apps/.

Appendix I – User Tasks and Questionnaire

After the tests conducted on DETI Interact, users were asked to fill the following questionnaire:

DETI Interact: Questionário do Utilizador

Instruções: Agradecemos a sua colaboração na realização deste estudo, que tem por objectivo avaliar a *Interface de Utilizador* do *DETI Interact* e, consequentemente, tentar melhorá-la seguindo os critérios de *Usabilidade*.
A sua colaboração constitui um factor importante para o êxito desta avaliação, por isso solicitamos-lhe o preenchimento deste questionário, cujos dados serão usados com total anonimato apenas para fins científicos.

1. Dados pessoais

Nº Mecanográfico: _____ Turma:_____ Data:_____/_____/_____
Género (M/F): _____ Idade: _____ Nº de Utilizador: _____

Tem experiência com dispositivos que possuem ecrã sensível ao toque, acelerómetro ou bússola digital (por exemplo *smartphones*)? S___ N___

Se respondeu Sim:

Que tipo de aplicações usa com as tecnologias referidas?
Navegação no sistema___ Escrita de mensagens___ Jogos___

Outras:_____

Usa: Várias horas por dia___ várias horas por semana___ várias horas por mês___
Várias horas por ano___ outro tipo de utilização:_____

2. Opinião geral sobre o *DETI Interact*

Após a utilização do sistema e tendo em conta a sua avaliação final, preencha o círculo que melhor reflecte a sua opinião em relação à utilização da ferramenta. Caso considere que estas quantificações não são aplicáveis, escolha NA.

2.1. Opinião sobre a utilização do *DETI Interact* *(preencha o círculo da opção que melhor corresponde à sua posição)*

2.1.1 Interacção através do **Smartphone:**

É fácil orientar-me no *DETI Interact*.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A velocidade de navegação é adequada.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA

A navegação é agradável.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A navegação tem algumas características irritantes.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A utilização do *DETI Interact* com esta forma de interacção é intuitiva.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A utilização do *DETI Interact* com esta forma de interacção requer treino.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A utilização do *DETI Interact* com esta forma de interacção é intuitiva.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA

2.1.2 Interacção através do Computador **Touch:**

É fácil orientar-me no *DETI Interact*.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A velocidade de navegação é adequada.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A navegação é agradável.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A navegação tem algumas características irritantes.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A utilização do *DETI Interact* com esta forma de interacção é intuitiva.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A utilização do *DETI Interact* com esta forma de interacção requer treino.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A utilização do *DETI Interact* com esta forma de interacção é fácil.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA

3. Por favor indique o grau de satisfação na utilização de cada uma das formas de interacção numa escala de 1 (muito baixo) a 5 (muito alto):

Smartphone:	1 nada satisteito	○	○	○	○	○	5 muito satisfeito

Computador Touch:	nada satisteito	○	○	○	○	○	muito satisfeito

4. Por favor assinale com qual das formas de interacção preferiu navegar no *DETI Interact*:

Smartphone: ☐ ☐ ***Touch***

5. Se pretender pode deixar aqui comentários sobre as formas de interacção usadas no *DETI Interact*:

_____
_____
_____
_____
_____

6. Que outros serviços gostaria de ver integrados no DETI Interact.

_____
_____
_____
_____
_____

FIM

Muito obrigada pela sua colaboração!

palavras-chave

Interacção Humano-Computador, Ecrãs públicos, Ecrãs de grandes dimensões, Usabilidade, Computação móvel

resumo

O uso de ecrãs de grandes dimensões para a exibição de conteúdo informativo contém limitações a nível de interacção que devem ser superados. Os dispositivos móveis actuais, equipados com acelerómetros, bússolas digitais, e ecrãs sensíveis ao toque, constituem uma alternativa para permitir esta interacção sem a necessidade de hardware adicional, uma vez que os utilizadores transportam o periférico consigo.

Esta dissertação apresenta os passos necessários ao desenvolvimento do DETI Interact: um sistema informático para exibição de conteúdo informativo para o Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro, que permite a utilizadores usarem um dispositivo móvel Android para interagir com os conteúdos apresentados. Após o desenvolvimento foram realizados testes com utilizadores e o resultado final foi colocado em funcionamento nos ecrãs de grandes dimensões presentes na entrada do departamento.

keywords

Human-Computer Interaction, Public Screens, Large Screen Displays, Mobile Computing, Usability

abstract

The use of large screen displays for the presentation of informative content has interactivity limitations that should be overcome. Current mobile devices come equipped with accelerometers, digital compasses, and touchscreens that constitute an alternative to allow interaction with these displays without the need for additional hardware, since the users will be carrying the peripheral with them.

This dissertation presents the necessary steps in the development of DETI Interact: an information system for the exhibition of informative content from the Department of Electronics, Telecommunications, and Informatics of the University of Aveiro, which enables users to use an Android mobile device to interact with the displayed contents. After its development, it was submitted to user testing and it is now operating in the large screens present in the department.

Table of Contents

Index of Figures

Acronym List

API Application Programming Interface

ASP Active Server Pages

CLR Common Language Runtime

CPU Central Processing Unit

DETI Departamento de Electrónica, Telecomunicações e Informática

DSD Distribuição de Serviço Docente

FCL Framework Class Library

GDI Graphics Device Interface

GPS Global Positioning System

GPU Graphical Processing Unit

HTTP Hypertext Transfer Protocol

IDE Integrated Development Environment

IIS Internet Information Services

LCD Liquid Cristal Display

LINQ Language Integrated Query

MFC Microsoft Foundation Class

MVVM Model-View-View Model

NFC Near-Field Communication

NUI Natural User Interface

OEM Original Equipment Manufacturer

OS Operating System

SDK Software Development Kit

UI User Interface

URI Uniform Resource Identifier

URL Uniform Resource Locator

UX User Experience

W3C World Wide Web Consortium

WCF Windows Communication Foundation

WPF Windows Presentation Foundation

WSDL Web Services Description Language

1 Introduction

1.1 Motivation

As technology evolves, computer and television screens have become increasingly thinner and cheaper. In addition, some of these screens are even equipped with touch-sensitive panels enabling interaction via touch input. This has allowed these screens to be used in public locations such as airports, train stations, malls, museums, lobbies of buildings, etc., presenting relevant information to the people passing by. The use of these screens, however, presents a challenge in how information is displayed and how people can interact with it. When the screens are touch-sensitive it is relatively easy to make the system interactive, however, when they are not, the system usually displays a predetermined set of information, usually presented in a looping or alternating sequence. The objective behind DETI Interact is to study how to make these screens interactive, allowing the user to control the flow of information and choose what he wishes to see.

Interaction with these screens can be achieved in a variety of ways, as can be seen in several projects presented in Chapter 2. These projects research the use alternative input peripherals, such as a smartphone, to more advanced scenarios such as video tracking or motion capture. DETI Interact is a follow-up project to DETI Guide (Palha 2010), a previous study that investigated how a smartphone could be used to interact with a public screen, for this reason, the chosen approach was to use a smartphone. The result is that there will be no hardware changes required on the system (for instance, no tracking cameras, touch panels, etc.), since users will be carrying the interaction peripheral themselves. However, the choice of how interaction is done is only one of the steps.

Given the nature of the device, several aspects concerning the interaction have to be addressed with care: the way the content is displayed on the screen, the way the user interface is designed, and the way the device interacts with the content, are all important factors, which will be addressed in this dissertation.

1.2 Objectives

DetiGuide(Palha 2010) was a system developed in the scope of a previous dissertation, which studied the possibility of using of a smartphone to interact with the screens in the lobby of the Department of Electronics, Telecommunications, and Informatics of the University of Aveiro. The present dissertation intends to go further. Following the results from the previous study, this project aims to develop of a platform for displaying interactive information on those screens, and integrate the possibility of interacting with this information via a smartphone.

This project has two main objectives:

- System architecture: the existing DSD (*Distribuição de Serviço Docente*) Platform which supplies information to the system that is currently operating in the lobby of the department should be studied in order to use the supplied information to develop interactive contents.

- Interaction with the displays: the developed system should allow a user in the lobby to interact with the screen to access relevant information, such class times, information regarding the teachers, etc. In addition to planning the system for communication with mobile devices, the system should also be prepared to communicate with other devices, such as a WiiMote or cameras via body tracking. It should also be prepared to run on computers with touch-sensitive screens.

The resulting developed system will be deployed to the computers in the lobby of the department where users will be able to interact with the system using their phones.

1.3 Dissertation Structure

This dissertation consists of five chapters, excluding this introductory section.

In chapter 2, “State of the Art”, the bibliographic research done for the development of this system is presented. It encompasses several projects that propose different methods of interaction between users and large screens.

Chapter 3, “System Design”, illustrates the different steps in the preparation of the development of the system. It describes the used tools, studies features that were chosen to integrate the system, as well as some of the required APIs, and presents the architecture of the system, leading to the next chapter.

In Chapter 4, “Development”, the development phase of the project is described. Some implementation details as well as problems are explained in order to illustrate how the core of the system works, and how it can be modified or extended.

Chapter 5, “User Evaluation”, explains how the developed system was tested. The results of the experiments are presented, and are discussed as to how they are used to improve the system.

Finally, in Chapter 6, “Discussion”, the work is reviewed, along with the results gathered from the testing sessions. Some possibilities for future work on the developed system are discussed.

2 State of the Art

2.1 Introduction

With advances in technology making computer screens increasingly thinner and cheaper, these became more common in public places such as shops, cafes, museums, atriums of buildings, and train and subway stations. However, these screens are often used to display information, rather than allowing users to interact with it. These public displays fostered several research projects to investigate interaction between users and the displayed information using mobile devices, augmented reality, and motion capture.

The projects studied for the development of DETI Interact propose diverse ways for interaction with information presented on computer screens. Apart from the many differences between these projects, some recurring elements allowed these studies to be grouped into three distinct categories. The first group of projects focuses the use of smartphones for interaction, concentrating on the technologies that can be used for communication between devices such as Wi-Fi, Bluetooth, and SMS. They present different methods of communicating data between mobile devices and large screens in order to interact with some part of the system. The second group of projects uses different approaches of video tracking and augmented reality to allow interaction between mobile devices and the target screens. Finally, the third group uses motion capture for the interaction, eventually dismissing the use of mobile devices altogether.

2.2 Communication between devices

Since DETI Interact will involve communication between a mobile device and a computer, the several means of achieving this communication must be studied, allowing for a more educated decision.

Gmote (Stogaitis and Sun 2008) is an example of an Android application that enables interaction between a mobile device and a computer, using a Wi-Fi connection. For the devices to communicate an application that acts as a server is deployed to the computer and a client application is executed on the mobile device. This system supports most of the media playback related features found in a standard remote control such as play, pause, rewind, volume control, etc. as well as new features like file browsing and media streaming. It can also use the mobile device's touchscreen as a touchpad, as well as Android's software keyboard as a keyboard, enabling the user to control the mouse pointer and keyboard remotely. Similar applications such as RemoteDroid, BoxeeRemote, TivoRemote, and SqueezeControl, have also been developed that allow interaction between additional mobile and desktop operating systems (Wired 2009).

While Gmote can be used to interact with a computer, similar applications have been developed that allow interaction with a television or a set-top box. Samsung has developed the Internet@TV platform (Samsung 2010), a service that allows users to access online content such as photos, video, music, social networks, news, games, etc., on an internet-enabled Samsung TV or Blu-ray player. Interaction with the system can be done with a regular remote control, however, Samsung has developed an application for iOS and Android-based devices, which lets users interact with the television or set-top box using their mobile phone. Google has also developed a similar system, Google TV (Google 2010), which also supports the use of a phone to control the device. In addition to typical functions of a remote control, Google's phone application also enables streaming video and audio content to the television.

Other systems, such as Touch & Interact (Hardy and Rukzio 2008; Hardy and Rukzio 2008), allow users to interact with large displays using mobile devices with NFC (Near Field Communication) technology. NFC consists of a set of short-range wireless technologies, such as RFID, typically relying on radio frequency for communication. The display used in this system is attached to a mesh of NFC tags, which enables the phone to detect the tags when close to the display. Given the nature of NFC tags, the system supports gestures such as hovering the phone over a certain tag, or touching an area close to the tag. The phone is used to identify the tags on the display, and communica-

tion with the system is done using Bluetooth. Figure 2.1 shows a user selecting cells on a map application on a display using Touch & Interact.

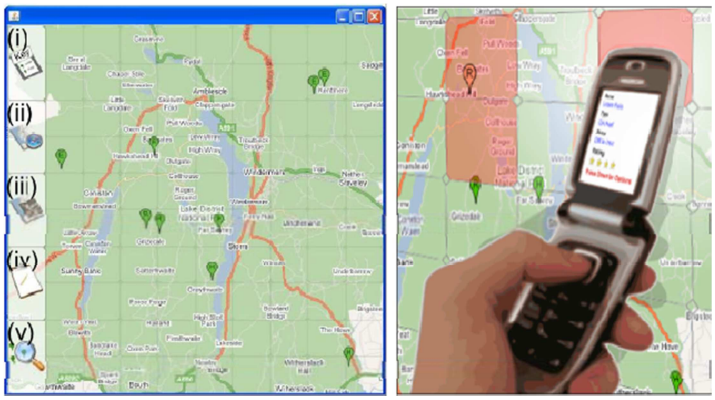

Figure 2.1 - A tourist guide application using Touch & Interact (Hardy and Rukzio 2008).

The principle behind Touch & Interact also produced two additional projects: Touch & Connect and Touch & Select (Seewoonauth, Rukzio et al. 2009). Touch & Connect allows users to transfer files between a laptop and an NFC phone simply by touching the NFC tag placed on the laptop. The system then configures the connection and transfers the date, without any additional user input required. For Touch & Select, a mesh of NFC tags similar to the one used in Touch & Interact allows the user to select items on the laptop by touching its display with the mobile device. Communication with the system in either case is done via Bluetooth.

MAGICBoard (Tang, Finke et al. 2008) is a public digital forum designed to encourage bystanders to interact with the system. Bystanders are individuals that are close to the large display but never actively participate in using the system. The system allows users to view or discuss on popular topics, and participate in some voting sessions. Interaction with the system was achieved by using either SMS messages, or a nearby kiosk terminal. An example of a deployed MAGICBoard application can be seen in Figure 2.2. Polar Defence (Finke, Tang et al. 2008) is a tower defence game that was deployed in a similar setting, using the same system behind MAGICBoard. Users would play the game by sending SMS messages to the system with the appropriate instructions. These projects attempted to reduce barriers to interaction, such as embarrassment, trust and security, and encourage spectators to become actors in the system (Kaviani, Finke et al. 2009).

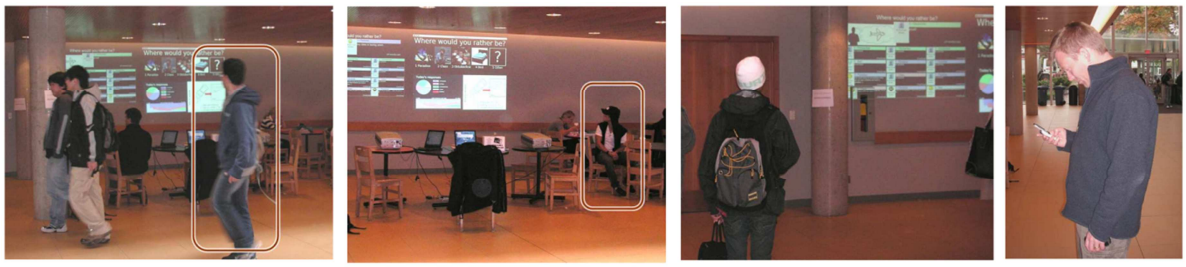

Figure 2.2 – A deployed MAGICBoard (Tang, Finke et al. 2008) system illustrating, from left to right, a *passer-by*, a *stander-by*, and *engaged bystander*, and a *contributor*.

As mentioned in Chapter 1, DetiGuide (Palha 2010), a project that preceded DETI Interact, studied interaction with large displays using a mobile device running Android OS. The system was deployed to computers located at the lobby of the Department of Electronics, Telecommunications, and Informatics of the University of Aveiro, and consisted of a web application similar to the department's website, and a mobile application, used to interact with the web application. The web application displayed information regarding how to navigate in the system through a series of widgets arranged along the edges of the web page. This information would assist the user in the manipulation of the mobile device in order to navigate on the web site. The mobile application supported several interaction techniques, such as using the accelerometer, gestures drawn on the touchscreen, and methods to control the mouse pointer. Communication between the mobile device and the web application was done over a Wi-Fi connection.

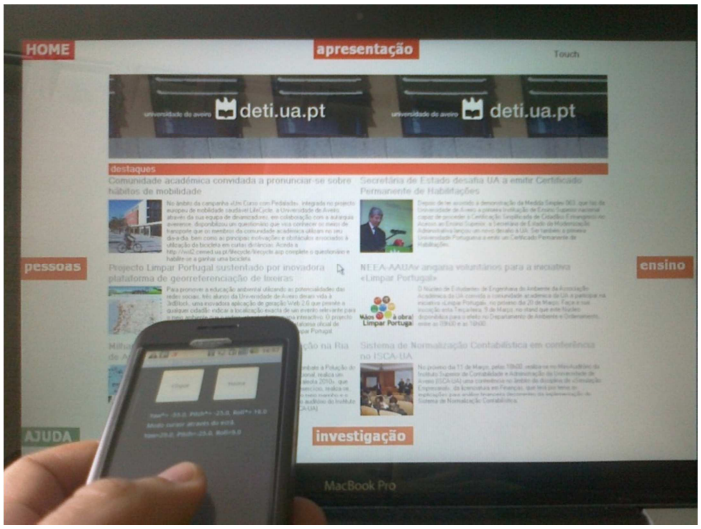

Figure 2.3 – User interacting with DetiGuide using the accelerometer on a mobile device.

In conclusion, the most used communication methods include Bluetooth, Wi-Fi, or SMS. For DETI Interact, Bluetooth and Wi-Fi present the most versatile alternative since there is no interference from the operator in the communication process.

2.3 Video Tracking

Interaction with large displays can also be achieved by tracking objects on a video feed, such as the movement of light sources. LasIRPoint (Cheng and Pulo 2003) uses an infrared tracking device and an infrared laser pointer in a presentation environment. The tracking device is coupled with the projector, facing the screen, so that it captures the gestures performed by the user with the laser pointer. Users can perform gestures such as selecting an object by drawing a circle around it with the laser pointer, and the system processes these gestures and proceeds accordingly.

C-Blink (Miyaoku, Higashino et al. 2004) uses a camera to track the movement of a mobile phone with a coloured screen (Figure 2.5). The phone runs an application that rapidly changes the hue of the LCD screen; this hue sequence is unique and easy to detect, and it can be used to identify multiple mobile phones, which allows the system to support multiple users. The camera is positioned above the display, and the user waves the phone in front of it (Figure 2.4). It uses the signal coming from the phone to track its position and uses this to control the position of the cursor on the display.

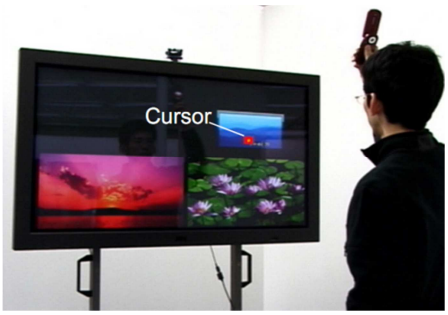

Figure 2.4 – Using C-Blink to control a display (Miyaoku, Higashino et al. 2004).

Figure 2.5 – C-Blink with an overlay of the video captured by the sensor (Miyaoku, Higashino et al. 2004).

In addition to identifying the user, a C-Blink signal can also encode extra information. To illustrate this, the system also implements several basic functions such as Click, Grab, and Pitch. Each of these functions is sent to the system through a C-Blink signal. For instance, if a user wants to download an image from the system (using the Grab function) he aims the device at the image and presses a button; this generates a C-Blink signal that encodes the terminal ID of the cell phone and along with a flag for the grabbing function. The system then responds by sending the image to the phone via the wireless network.

Touch Projector (Boring, Baur et al. 2009) is a system that uses an augmented reality solution allowing users to remotely manipulate content presented on remote displays on a mobile device via a live video image captured by the device's camera. The user aims the mobile device at a display and interacts with the image on the screen via touching and dragging the objects. Input to the touchscreen is "projected" to the selected display as if it had occurred there. The user can select an object and move it with a finger (Figure 2.6), which results in a relatively precise movement, or he/she can select and hold an object and move the device instead, which performs a more coarse movement. Objects can also be moved between displays by touching the object in the first display and moving the device towards the second, dragging the object off-screen and into the intended display.

Figure 2.6 – Selecting an object using Touch Projector (Boring, Baur et al. 2009).

In order to achieve a better user experience, features such as zooming and freezing the video image are also available. Zooming allows the user to interact with distant displays; this can be done automatically when the user aims at a display where the image does not fill the whole area of the video. Freezing the image allows for higher precision since the user will be dealing with a still image; it also eliminates the need to keep the device still or pointed at the screen, which avoids fatigue.

Madhavapeddy *et al.* (Madhavapeddy, Scott et al. 2004) also present a camera-based interaction using mobile phones, involving manipulating tagged interactive UI elements such as sliders and dials (Figure 2.7). Interacting with the system relies on using *SpotCodes* (Figure 2.8), a visual tag similar to a QR-Code, which is detectable by the camera-phones. These tags can be active (e.g. generated dynamically by the system) or passive (e.g. printed on paper). By placing the phone close

to the tag, ensuring the camera on the phone can identify it, the user can position a graphical slider, or orient a graphical dial by manipulation the camera-phone.

The mobile phone is responsible for tracking, and identifying the tags, while the system is responsible for displaying and refreshing the tracked images. Communication between the devices is achieved using Bluetooth.

Figure 2.7 – Madhavapeddy's *et al.* (Madhavapeddy, Scott et al. 2004): User interacting with a world map application.

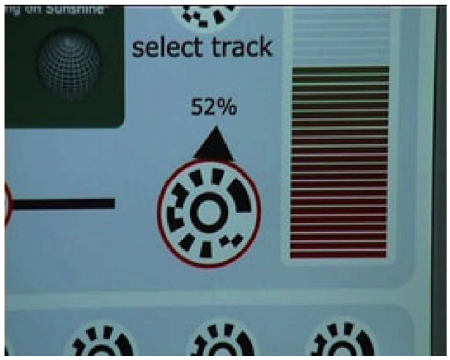

Figure 2.8 - Example of a dial tagged with a *SpotCode* using Madhavapeddy's *et al.* (Madhavapeddy, Scott et al. 2004) system.

2.4 Motion Capture

Other ways used to interact with information displayed on large screens include motion capture techniques. Motion Capture systems typically require specific cameras, such as Infrared or Stereoscopic cameras, as well as markers on the tracked objects to assist in the motion capture process.

Vogel *et al.* (Vogel and Balakrishnan 2004) developed a prototype system for use in interactive public ambient displays which allows for multiple users. Interaction with the system is performed using simple hand gestures (Figure 2.9) as well as touch screen input. A motion capture system is responsible for the detection of users and gestures, and requires the use of small passive markers on the body parts that require tracking. While this is inconvenient, advances in computer vision techniques should obviate the need for markers.

Figure 2.9 – Users interacting with the prototype system developed by Vogel *et al.* (Vogel and Balakrishnan 2004).

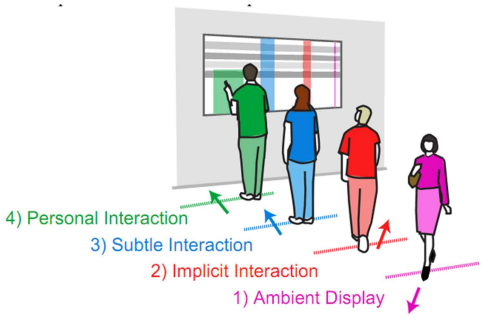

Figure 2.10 – Illustrating the four interaction zones covered by the system(Vogel and Balakrishnan 2004).

This prototype explores the concept of transitioning from implicit to explicit interaction with both public and private information. The system covers four interaction zones: Personal Interaction, Subtle Interaction, Implicit Interaction, and Ambient Display, as shown in Figure 2.10. When a user is far from the screen, the system will perform as an ambient information display, showing a range of categorized information. If the system detects that the user is closer and facing the screen, the system enters the Implicit Interaction phase, adapting the UI by displaying brief information concerning the user, as well as subtle notifications regarding personal or public information items that may require attention. As the user approaches the screen, the system enters the Subtle Interaction phase; the system adds more detail to the information and notifications onscreen. When the user selects an information item the system enters the Personal Interaction phase, the user is now in front of the screen, which allows for direct manipulation of the information.

Underkoffler (Underkoffler 2010) developed a system in which users would wear a set of gloves with reflective material placed on the tips of the fingers. The system would then be able to track the movement of each finger, which would allow the user to interact with large displays using hand gestures. This system was developed to be used on the sci-fi movie Minority Report, and has been updated to allow interaction with three-dimensional environments.

Orange, the UK based mobile network operator, unveiled a gesture based interaction screen, developed by The Alternative (TheAlternative): the Orange Interactive Window (GIZMODO) (Figure 2.11). The system uses cutting-edge motion capture technology that tracks gestures executed by the users to control the content on the screen without the need to place passive markers on the user (Figure 2.12). Other competitors in the mobile market such as Vodafone, Samsung, and Sony, have also developed their own versions of this interactive window.

Figure 2.11 – The Orange Interactive Window (GIZMODO): Main display.

Figure 2.12 – User interacting with the Orange Interactive Window (GIZMODO).

Kinect (Microsoft 2010), developed by Microsoft, also offers state-of-the-art motion capture technology. It is intended for home use on the Xbox 360 video game platform and eventually computers equipped with Microsoft Windows (Anthony 2010).

Figure 2.13 – The Kinect sensor device (Microsoft 2010).

The device, shown in Figure 2.13, includes an RGB camera, a depth sensor and a microphone array (Totilo 2010). The RGB camera is used as a regular video camera, and can be used for facial recognition. The depth sensor consists of an infrared laser emitter and a CMOS Active-Pixel Sensor. The emitter projects a mesh of infrared points to the surrounding environment, which the sensor captures; the distance between these points is used to tell how far a certain object is. By mapping the depth image with the image from the RGB camera, the system can create a 3D video image of the surroundings. This allows the system to provide 3D full-body motion capture under any ambient light conditions. The microphone array can suppress ambient sounds, and locate sound sources, which enables innovative voice recognition capabilities.

Figure 2.14 – Using Kinect with an application in Windows 7.

Figure 2.15 – Two users interacting with Kinect simultaneously.

The hardware behind Kinect along with its software technology enables the system to track up to six people simultaneously, including two active users (Figure 2.15). The system can recognize 31 body parts per user, which allows for the detection of gestures of a certain complexity. Since launching the device, Microsoft has revealed plans to integrate Kinect with Windows (Figure 2.14) (Anthony 2010), and has released a beta version of its software development kit to the public (Microsoft 2011) enabling developers to create Windows applications that target the Kinect device. This differentiates Kinect from its competitors developed by Sony and Nintendo.

3 System Design

In this chapter, the choices dealing with the definition and the design of DETI Interact will be detailed. The content to be presented by the system will be specified, as well some requirements to present it. This will also lead to the evaluation of development platforms. Finally, the system architecture will be explained.

3.1 Introduction

The Department of Electronics, Telecommunications, and Informatics (DETI) of the University of Aveiro has, in its lobby, three large screens. These screens display a website that contains information related to the department, specifically the most recent news, and a list of all the lecturers in the department. The information contained on these webpages is stored in a server running the DSD Platform (*Distribuição de Serviço Docente*).

DETI Interact intends to replace the existing system by displaying additional information and allowing users to interact with it, enabling navigation and selection operations in the system. Interaction with the system will be done via mobile devices powered by the Android operating system. Communication between devices must be achieved seamlessly by the system on behalf of the user.

In order to conduct a better evaluation of the system, interaction with the system with a mobile device will be compared with interaction with a touchscreen, as well as a mouse. To ensure some fairness in the comparison, the system will include different types of content and methods to interact with it that better leverage the advantages and disadvantages of each device. This content includes information from the university's servers (regarding lecturers and timetables), as well as ma-

nipulation of a map from Google Earth, and three-dimensional content. Each of these sources of data has its own method of integration with the system, such as a dedicated API (Application Programming Interface) or a series of Web Services. This requires that each source be studied as to how it can be included in the system, which will in turn affect the development environment, since some APIs may be exclusive to a certain platform or programming language.

3.2 System requirements

The selected content to be presented on the initial version of DETI Interact includes content from the department's DSD Platform, as well as a map from Google Earth, and three-dimensional content. The DSD platform holds information relevant to the department, thus it will be the source for this data. From the DSD server information regarding the department's teachers and timetables will be gathered. Google Earth offers an intuitive way to use the touchscreen on the mobile device, via gestures like scrolling or pinch to zoom, making it a fitting addition to the system. The use of the digital compass and the accelerometer on the mobile device brings an advantage to the presentation and manipulation of three-dimensional content making it another suitable addition.

The DSD Platform, Google Earth, and existing 3D APIs must be studied to determine how these can be implemented in DETI Interact.

3.2.1 DSD Platform

The DSD Platform (Campos 2008) is a system that allows lecturers and students to perform operations related to the academic year. It is actively used every year by students on situations such as checking timetables, and enrolling in subjects. Students can also make choices regarding dissertation proposals, which are added to the system by the lecturers. It is, essentially, a management system for the department.

The system consists of a server computer running a Microsoft IIS (Internet Information Services) web server, which is hosting an ASP.NET web application that allows remote access to the system. The main webpage for this application can be seen in Figure 3.1. It also contains a SQL database managed by Microsoft SQL Server. The database holds all the data relevant to the department such as lists of students and lecturers, all the lectured courses along with the corresponding year and semester, class times and classroom occupation, etc.

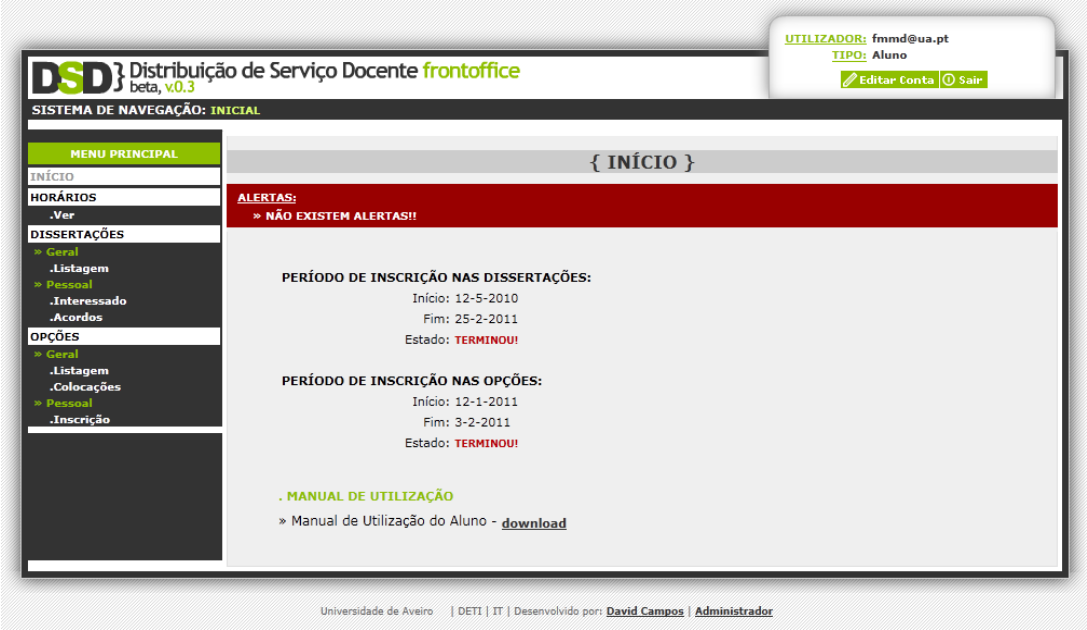

Figure 3.1 – The DSD Platform's web page.

Some of the information in the database can be accessed via a series of Web Services published by the ASP.NET Web Application. A Web Service is, as defined by the W3C, "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards" (W3C 2004).

Web services allow programmers to easily access the DSD platform's API, meaning that DETI Interact can collect and handle data from the server. The existing Web Services are quite limited in what concerns to the type and amount of information they return, but they provide enough data for some basic operations. The data returned can be used, for example, to recreate the existing list of teachers, as well as evaluate classroom occupation, generate timetables, etc. Furthermore, should these Web Services be insufficient, new ones can later be developed if the need arises. The list of available Web Services is as follows:

- getAreas();
- getAulas();
- getCursos();
- getDepartamentos();
- getDisciplinas();
- getDisciplinasAno();
- getDocentes();
- getDocentesAno();

- getSalas();
- getSemestres();
- getSubAreas();
- getTiposSala();
- getTurmas();
- getTurmasHora();
- getWishList();

3.2.2 Google Earth

Google Earth (Google 2005) is a virtual globe application, with mapping and geographical features. It maps the earth's surface by overlaying images obtained from satellite imagery and aerial photography. The product was initially released as a standalone, downloadable application. However, it has since become accessible via a web browser plugin.

Introducing Google Earth to DETI Interact provides a way of comparing different methods of interaction. Given its graphical user interface, it constitutes a suitable choice to compare interaction with mouse and keyboard, with interaction using a touch screen on a mobile device. Google provides APIs for developers to use its mapping services, making integration of the Google Earth plugin possible. This API, however, is only accessible through Javascript, which is intended for use on web applications. Including Google Earth on the system involves finding a way to parse HTML and Javascript and displaying the results within the application. This can be done by either accessing Web Browser controls or using libraries on the OS for that purpose.

3.2.3 3D Content

Manipulation of three-dimensional content represents another appropriate choice for the comparison of different interaction devices. The use of compass or accelerometer on mobile devices can be compared with the use of the mouse and keyboard, or a touchscreen, on a computer.

There are numerous ways to display three-dimensional content on a computer, including OpenGL, DirectX, VTK, and XNA. These graphical APIs are used to produce standalone applications, but can also be used to display three-dimensional content within applications, albeit with some minor adjustments. The use of any graphics API will depend on the development platform: OpenGL and VTK are cross-platform, DirectX is restricted to C++ development on Windows, although there are managed wrappers for .NET, and XNA is a Windows exclusive as well.

In order to use any of these APIs to render a scene within a desktop application it is necessary to either use an available wrapper, which allows the developer to create a user interface around the three-dimensional content, or redirect the target render buffer to a different container (e.g. an image), and display this container in the application.

3.2.4 Mobile Devices

In order to target mobile devices it is necessary to decide whether to develop a web application, enabling all the mobile platforms to interact with the system, or to develop a native application, which targets a specific platform.

The ideal solution would be to develop a Web Application for mobile devices instead of a native application since it would be able to target any mobile device with a web browser, regardless of operating system type or version. It also provides better control over the application: there is only one application to develop and update, and every user will always be using the latest, most up-to-date version. However, the problem with web application would be accessing the device's hardware (Fling 2009). For DETI Interact, apart from the touchscreen, it is also useful to have access to hardware such as the accelerometer and Bluetooth adapter. In addition, access to the camera, GPS, storage, etc., might be necessary in future versions of the application.

Another solution for distributing for multiple platforms would be using PhoneGap (PhoneGap 2008). This mobile framework currently allows development for six different mobile platforms: Apple iOS, Google Android, Blackberry OS, Palm webOS, Windows Mobile, and Symbian. It provides access to hardware such as the accelerometer, camera, compass, GPS, and storage, but still offers no access to the Bluetooth adapter. Navigation in the applications built using PhoneGap also are not consistent with the operating system, for example, pressing the 'back' button on an Android device exits the application, instead of navigating back to a previous screen.

Since the mobile application for DETI Interact requires access to hardware unavailable when developing for both the mobile web and PhoneGap, the remaining solution is to develop a native application. Given that the Global market share for the Android platform has revealed a rapid growth and a high adoption ration, quickly becoming the leading mobile platform (DailyTech 2011) this was the targeted platform for the initial version of DETI Interact. In addition, there was already Android-powered hardware from previous studies, available to be used for this system.

Developing for Android is done via a provided SDK that allows access to all the required hardware, which is unavailable or limited in other platforms such as iOS or Windows Phone 7. Other mobile platforms can easily be addressed at a later stage.

3.3 Development Environment

With the specification of which content is to be presented, steps can be taken into the development of DETI Interact. Given that the DSD Platform is developed using the .NET framework, and Android devices require the use of the Java platform, the use of either of these environments to develop DETI Interact would be beneficial given the vast amount of existing libraries for each platform. Content such as Google Earth and 3D scenes can be easily included in an application developed in any language.

Since the target for the mobile application is an Android device, the Java platform could also be used to develop the application that will be running on the large screens. One big advantage of using Java would be its platform-independence, since the Java Virtual Machine is deployable to most known platforms. Being such a widespread platform, a vast amount of libraries, wrappers, and documentation is also available. Additionally, the Java Class Library includes a Bluetooth API, which would be useful for communication between computers and Android devices. On the other hand, while its performance has improved substantially over time, Java still displays poor performance in some scenarios, as well as a high consumption of system resources (Mihailescu 2010). Furthermore, regarding security, Java has become the most common target for malware, surpassing Adobe Flash and Adobe Reader (Cull 2010). This could represent a security issue since the system is to be deployed at a public location.

An alternative for the Java platform is the .NET Framework (Microsoft 2009). While it cannot match Java in platform independence, .NET still covers a wide range of platforms thanks to the Mono Project that implemented its own version of the Runtime; Mono's implementation of the framework, however, lags behind Microsoft's in some areas. The framework developed over time and now includes features unavailable on the Java platform such as Language Integrated Query, for data processing, and Windows Presentation Foundation, an innovative tool used for user experience and user interface design. In terms of performance, while .NET (and managed applications in general) will probably never be as fast and memory efficient as their native counterparts, applications tend to be faster, due to its Just-In-Time Compiler, and less resource intensive, due to its Garbage Col-

lector (Mihailescu 2010). In Addition, .NET assemblies also take advantage of Windows' Data Execution Prevention and Address Space Layout Randomization, which improves the overall security of the system (Richter 2010).

While communication via web services is independent of platform or language, consuming .NET Web Services from a different platform, such as Java, is not as streamlined as doing so from within the .NET platform. Google Earth and 3D content can be implemented in either platform. Furthermore, communication between the Android device and the desktop application via Bluetooth is done through the exchange of packets of data, such as OBEX messages, which is independent of platform or language. Taking in to account the pros and cons of each alternative, the chosen development environment for the computer application was the .NET Framework.

3.4 Development Tools

3.4.1 Software

In order to develop DETI Interact, two applications must be developed: a desktop application, in .NET, and an Android application in Java. Various tools were used to assist and speed up the development of these applications.

3.4.1.1 Microsoft Visual Studio 2010

Since the system would be built on the .NET platform, using Visual Studio was an understandable choice. Microsoft Visual Studio 2010 is the most recent version of this IDE (Integrated Development Environment). It ensures quality code throughout the entire application lifecycle, from design to deployment, and assists in the development of applications for Windows, Windows Phone, SharePoint, Web, Cloud, etc.

Visual Studio was used to develop the portion of the system that ran on the computers at the lobby of the department. The application along with some addition libraries was developed in C#, and the user interface was built with XAML, used by Windows Presentation Foundation and Silverlight.

3.4.1.2 Microsoft Expression Blend 4

Expression Blend is a user interface design tool belonging to Microsoft's Expression Studio suite. It is an interactive front-end for designing XAML-based interfaces for Windows Presentation Foundation, Silverlight, and Windows Phone applications (Microsoft 2011). Blend allows developers to focus on the user interface design of the applications, without affecting the business login layer. It promotes the usage of the MVVM architectural pattern.

3.4.1.3 Eclipse

Eclipse is a multi-language, multi-platform software development environment. It consists of an Integrated Development Environment and an extensible plug-in system. This plug-in system allowed for the development of the Android Development Tools (ADT) plug-in which includes a device emulator, and tools for debugging, and memory and performance profiling. It is the IDE recommended by Google, and was used for the development of the Android application.

3.4.2 Hardware

DETI interact is expected to run on a computer connected to a large display as well as a computer equipped with a touchscreen. These are both Asus machines: an Eee Box, connected to the large display, and an Eee Top. The mobile application that will interact with the system will be tested on a provided Motorola Milestone.

3.4.2.1 Asus Eee Box

The Asus Eee Box is a small form-factor desktop computer. It has a low-end processor and integrated graphics card, which allows it to consume very little power. This, however, can have a negative impact on the performance of the application, since the system will deal with very large sets of data, as well as three-dimensional content.

3.4.2.2 Asus Eee Top

The Asus Eee Top is a computer similar to the Eee Box line, but its hardware is built into a touch-sensitive screen. In addition to the low-end hardware, the screen only supports single-point interaction, which will limit the set of gestures that can be used as well as affect the overall user experience.

3.4.2.3 Motorola Milestone

The Motorola Milestone is a quad-band version of the Motorola Droid, an Android-powered smartphone. The hardware is similar between the two, with only minor differences in internal memory and installed applications. It was the device used to test the developed mobile application, and was also used for the tests performed with users. During the time the DETI Interact was deployed to the computer in the lobby, users were able to install the Android application in their personal devices.

3.5 DETI Interact Design

With all the system requirements taken care of, a development environment chosen, and all the hardware detailed, the system architecture as well as some design and development considerations will be described.

3.5.1 Architecture

As discussed previously, DETI Interact will consist of two applications. The first application will run on the desktop computers connected to the large displays. This application will contain the content gathered from the DSD server, the map from Google Earth, and a viewer for 3D content. To allow interaction from Bluetooth devices, a Bluetooth socket will be waiting to acquire a device. The desktop application will interpret data packets sent by the mobile device in order to perform operations such as changing the displayed content. While no device is connected to the system, the desktop application will be able to enter a demonstration mode, cycling through all the available content.

The second application will run on an Android mobile device. It will observe gestures drawn on the touchscreen, as well as the values from the digital compass or accelerometer, and send this data as a structured packet via Bluetooth to the desktop application. A diagram of the system architecture can be seen in Figure 3.2.

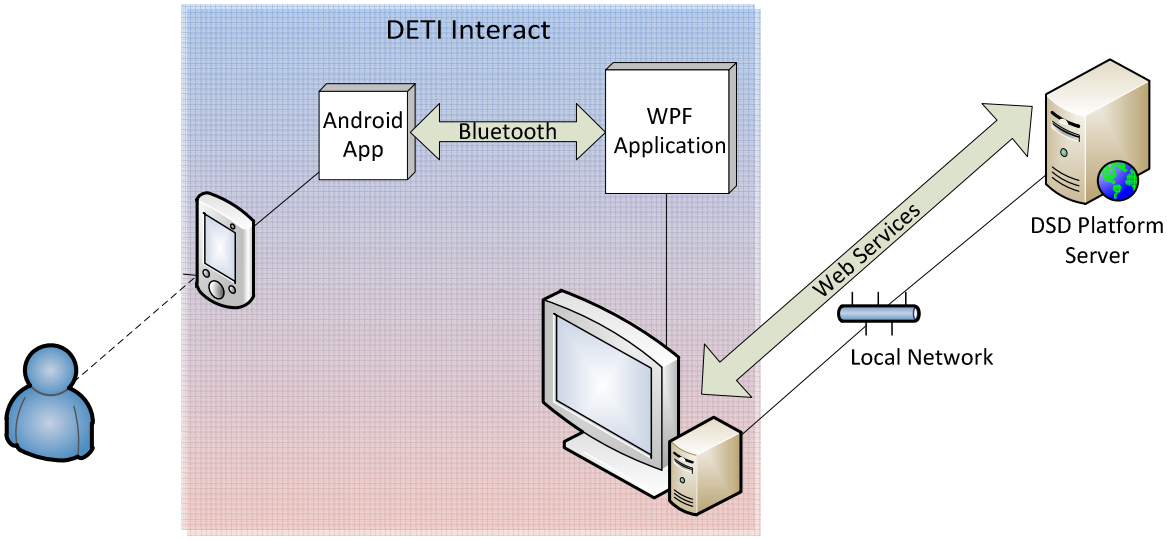

Figure 3.2 - DETI Interact architecture

3.5.2 Desktop Application – .NET Framework

As discussed previously, the .NET Framework was chosen as the development environment for DETI Interact. The mobile application, developed for Android devices, will be written in Java, and the desktop application, which will run on the computers on the department, will be written in C#.

The .NET framework is a programming environment developed by Microsoft for building, deploying, and running applications and services that use the .NET technologies. The framework consists of two main components: the Common Language Runtime (CLR) and the Framework Class Library (FCL). The CLR is a runtime that contains features such as memory management, assembly loading, security, exception handling, and thread synchronization, and is usable by various programming languages. Currently dozens of languages can target the runtime of which the most widely used is C#. This was the programming language used in the development of DETI Interact. It also includes features that are not available in other platforms but are useful for this system, such as LINQ and WPF.

LINQ – Language Integrated Query

Using .NET benefits DETI Interact when it comes to handling data from the DSD database due to the inclusion of LINQ. This feature adds native querying capabilities to .NET languages meaning

that the programmer can write code similar to SQL queries in order to parse data, instead of looping through all the results returned from the web services. This can speed up certain portions of the code, ensuring a more responsive application.

WPF – Windows Presentation Foundation

Another noteworthy addition to the .NET Framework is Windows Presentation Foundation (WPF). Microsoft describes WPF as being "a next-generation presentation system for building Windows client applications with visually stunning user experiences" (Microsoft 2010). Instead of depending on the older GDI subsystem like Windows Forms or MFC, WPF uses DirectX, which allows Windows to offload some graphics tasks to the GPU, enabling the CPU to work on tasks that are more significant. WPF also introduced a new architectural pattern: Model-View-ViewModel (MVVM). This pattern was created to make use of specific features in WPF and Silverlight, separating the View layer from the rest of the pattern. This separation of layers means that the business logic of the application is completely independent of the user interface.

3.5.3 Mobile Application – Android

As described on 3.2.4, the targeted mobile platform was the Android platform. Android is a software stack for mobile devices that includes an operating system, middleware and key applications (Google 2011). An SDK (Software Development Kit) is provided with the tools and APIs necessary to develop application on the Android platform.

Development for Android is done in Java, however, there is no Java Virtual Machine, instead the code targets the Dalvik virtual machine, which was specifically designed for Android and optimized for battery-powered devices with limited memory and CPU. The system integrates a web browser based on the WebKit engine, SQLite for data storage, and support for common audio, video and still image formats. It also features GSM telephony, Bluetooth, EDGE, 3G, WiFi, Camera, GPS, compass and accelerometer. All of these features are available in the platform's SDK.

The provided SDK allows developer access to the gestures drawn on the touchscreen, data from the compass and accelerometer, and the Bluetooth adapter. These are the necessary features for the system to work properly.

3.5.4 User Interface and User Experience

Designing a good user interface for the system and planning a good user experience are crucial to increase interest and to encourage users to use the system. The user interface is what allows the user to manipulate the application. It must provide effective ways for a user to operate and control the system. This should be done while providing a good user experience. User experience is a subjective concept since it varies between users. The system must be easy and designed in a way that the user does not feel the using the system is a chore. The user must feel that he/she knows how to operate the system at first contact, and can anticipate how the system will respond to his input.

There will be two very distinct screens in use: the large screen, and the mobile screen. This means that some care must be taken in how the screen will be used. Depending on how the screens display information, using both screens may cause user confusion. In order to avoid having the user diverting attention between the large screen and the mobile screen, the chosen approach was to have all the content presented on the large screen, and use the mobile device solely for interaction. This will allow the user to focus on the large screen, and forget about the mobile device, reducing the cognitive load on operating the mobile device correctly, in an attempt to resemble what happens with the computer mouse nowadays.

One other very important detail is how the information will be displayed on the screen. The available space must be used properly and it must provide a decent usage experience. Several aspects must be taken into account:

- The system will also be deployed to a touchscreen computer, which means that interaction with content on the screen must be finger-friendly;
- The large display is fixed on a wall, meaning that users will stand further away as they would from a regular computer, hence, content must be displayed in a larger format;
- Both screens have a widescreen format (with a 16:9 image ratio), with their width being almost twice as large as their height. Given the more limited height, screen space would be better used, for example, by stacking controls in a column rather than in a row (Figure 3.3), this, however, limits the amount of controls that can be stacked;

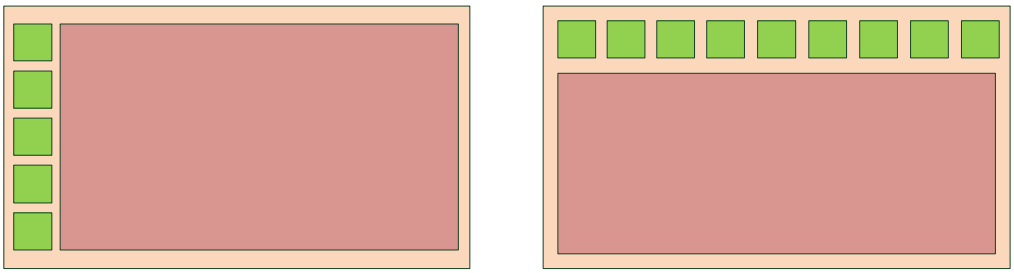

Figure 3.3 – Two distinct ways of using available space on a widescreen display.

- The screens will be manipulated by either mobile device or a touch screen, therefore the content, and interaction with it, must support a limited set of gestures.

Figure 3.4 - The original Xbox 360 dashboard (Thurrott 2010)

There are numerous ways of displaying content properly, each with their advantages. Media centres and videogame systems provide great user interfaces, directed for controllers with limited functionality namely directional and selection buttons. These systems provide a good starting point in the development of the user interface. The original dashboard included with the Xbox 360 video game system (Figure 3.4) provides a large amount of screen space for information, with different panes that can be used to switch context. This dashboard is derived from the UI used in Windows Media Centre, and newer versions of this dashboard have eventually blended with it. The concept behind these user interfaces has been in constant evolution for over a decade culminating in devices such as the Zune player and the Windows Phone (Thurrot 2010).

The original Xbox dashboard served as inspiration in the design of the user interface for the desktop application of DETI Interact. With information arranged in a similar manner a greater portion of screen can be used to display information, while the edges show which other panes area accessible.

3.6 Conclusion

In order to make DETI Interact a system appealing to the user, a series of features were chosen to integrate it. Communication with the DSD platform is required to provide content relevant to the department, and can be used to populate the system with a list of teachers, timetables, etc. In addition, a Google Earth plugin and the exhibition of three-dimensional content would make the system more engaging, as well as providing interesting methods for interaction with a mobile device.

The system will include two applications, a desktop application, which will be developed in .Net, for its performance, security, and included features, while the mobile application will be developed in Android, using Java, since this is the standard approach.

4 Development

In this chapter, both the mobile and the desktop application will be described in detail. In addition, some implementation details will also be presented.

4.1 Introduction

As mentioned, DETI interact is a system that uses two different applications: a mobile application, running on an Android device, and a desktop application running on a computer, which displays information on a large screen.

The Android application is tasked with collecting different gestures performed by the user. With no information being displayed there, the user can focus on the large screen, and use the mobile device merely as another peripheral.

The desktop application is responsible for displaying the information, as well as interpreting the gestures from the mobile application, and responding accordingly. This required a series of libraries to be developed. These were needed for fetching and processing the data from the DSD server, to enable the inclusion of and XNA application, and to interpret the gestures performed by the user.

4.2 Android Application

To interact with the desktop application a mobile application running on the Android operating system was developed. This application connects to a DETI Interact computer and transmits data through a Bluetooth connection. A Bluetooth service was developed to perform all Bluetooth-

related operations. In addition, the Android SDK supplies a `GestureDetector` class that was used for the detection of gestures. The values from the digital compass are also sent via Bluetooth to the computer.

4.2.1 Bluetooth Service

The `BluetoothService` class controls the communication with the desktop computer. To perform the connection, the Bluetooth MAC address of the computer must be known. This information is gathered when the devices are paired (as stated in 4.3.2.3, Android's Bluetooth API enforces device pairing). When the application is launched, a list of connectable Bluetooth devices is displayed. Once the user chooses a device, its address is passed to the `BluetoothService` class, which then tries to connect to it. If the connection succeeds, the input and output streams are initialized, enabling the application to read and write from them.

A `write()` method is supplied by this service that can be used to place an array of data into the output stream of the Bluetooth connection. The streams between two Bluetooth (or Serial) endpoints can be written or read at any time, by different clients. If two clients were to write at the same time, whoever reads the stream afterwards will find a mixture of both messages. In order to avoid this situation, the provided `write()` method only allows one message to be written at any time. Whenever the system needs to write some data to the stream, the `write()` method blocks any further calls, in order to avoid conflicts. While the method is blocked, any awaiting messages are discarded, since the system is expected to respond to user input in real time, and answering these messages would result in the system responding with some lag.

4.2.2 Main Application

The main activity for the Android application requests a Bluetooth connection to the aforementioned service. The information collected by the sensors and sent to the Bluetooth services is placed in a string, with a predetermined size and structure. This structure, shown below, identifies the gesture, as well as three distinct fields, for the X, Y, and Z components of a scroll or fling, or the Yaw, Pitch and Roll components from the device's orientation. Unused fields are left with a null value.

Bytes:	0	1-2	3	4-12	13	14-22	23	24-32	33
Content:	"["	Gesture ID	":"	Value 1	":"	Value 2	":"	Value 3	"]"

This structure has a fixed size to aid the computer in the decoding. This eliminates some problems that existed on an initial version of the system where the computer was decoding messages at a slower rate than they were being sent. This caused some conflicts in the messages such as messages cut in half, or concatenated messages, which interfered with the gesture detection.

Concerning the detection of gestures on the device's touchscreen, the `GestureDetector` class offered by Android's API identifies most of the needed gestures such as Tap, Scroll, and Fling. However, the Pinch gesture, used to zoom in and out, is unavailable in the API for Android 2.1. A `ScaleGestureDetector` class was introduced in version 2.2 of the API to allow the detection of pinch to zoom gesture.

This constitutes one of the main problems of the Android platform: fragmentation. Mobile operators and OEMs (Original Equipment Manufacturers) load devices with their software, or modified firmware versions, which have to be adapted when Google launches a platform update. New features, such as the `GestureScaleDetector`, are added to these updates, and are not backwards compatible. Most devices will only receive these updates later if at all. The result is that the Android ecosystem includes devices running android versions from 1.6 up to 3.0. If an application uses a feature from the API from version 2.2, every device running Android 2.1 or under will not be able to run the application. For this reason, and since development was started when Android 2.1 was the most recent Android version, the zoom gesture had to be implemented from scratch. This is done by detecting two fingers on the device's touchscreen, and accompanying the variation in spacing between the fingers.

4.2.2.1 Supported Gestures

The set of gestures currently supported by the system include Scroll, Fling, Tap, Long Press, Rotation, and Zoom.

Scroll: scrolling (Figure 4.1) is done on the mobile device by touching a point on the device's touchscreen, and dragging the finger to another position. Any direction on the XY axis is valid and therefore transmitted to the desktop application.

Fling: the fling gesture (Figure 4.2) is a quick finger swipe, across the device's screen, in any direction. It is usually used to perform an accelerated scrolling gesture, known as Kinetic Scrolling.

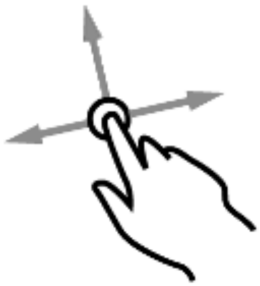

Figure 4.1 – The Scroll Gesture (Ideum 2011).

Figure 4.2 – The Fling Gesture (Ideum 2011).

Figure 4.3 – The Tap Gesture(Ideum 2011)

Tap: a tap (Figure 4.3) consists of touching the screen for a short duration of time. It is usually used to perform selections.

Long Press: a long press (Figure 4.4) is done by touching the screen for a longer period of time (about one second), which differentiates itself from the tap gesture, as well as accidental touches. This gesture is usually used to trigger the display of a context menu.

Pinch to zoom: zooming (Figure 4.5) is done placing two fingers on the touchscreen and moving them closer together or apart, to zoom out or in, respectively.

Figure 4.4 – The Long Press gesture (Ideum 2011).

Figure 4.5 – The Pinch to zoom gesture (Ideum 2011).

Figure 4.6 – The Rotation gesture.

Rotation: the values from the device's digital compass are constantly polled and sent to the desk-top application. These values are used to detect several rotation gestures (Figure 4.6).

The development of the desktop application can be targeted to take advantage of this gesture library.

4.3 Desktop Application

As discussed in section 3.2.1, the Web Services available in the DSD platform, albeit limited, are enough to list lecturers and generate schedules. These two features were chosen to integrate the first version of the application. In addition, since current mobile devices are equipped with multi-touch displays, accelerometers and digital compasses, a map and a three-dimensional object viewer were also included in the application.

During application development, attention was taken to ensure system extensibility, so that it would be easy to add, remove, or modify core features.

4.3.1 Structure

The desktop application was developed in C# using Windows Presentation Foundation (WPF) for the user interface, using the Model-View-ViewModel pattern where appropriate. This prompted for the structure shown in Figure 4.7.

The application presents a user interface with a series of pages, with different content being presented on each page. In the initial version of the application, there is a page for the list of teachers, a page for the timetables, a page for the Google Earth plugin, and a page for the 3D model viewer. There is also an additional page where help is presented to the user. Every page hosts a WPF control, which is tasked with displaying the specific content, and offers several interaction possibilities. This is achieved by using the developed `DetiControl`. This control provides a series of properties, methods, and events so that the application can control its operation. By adding a `DetiControl` to the application, a new page is automatically created on the UI and it will operate in the same way as any existing control.

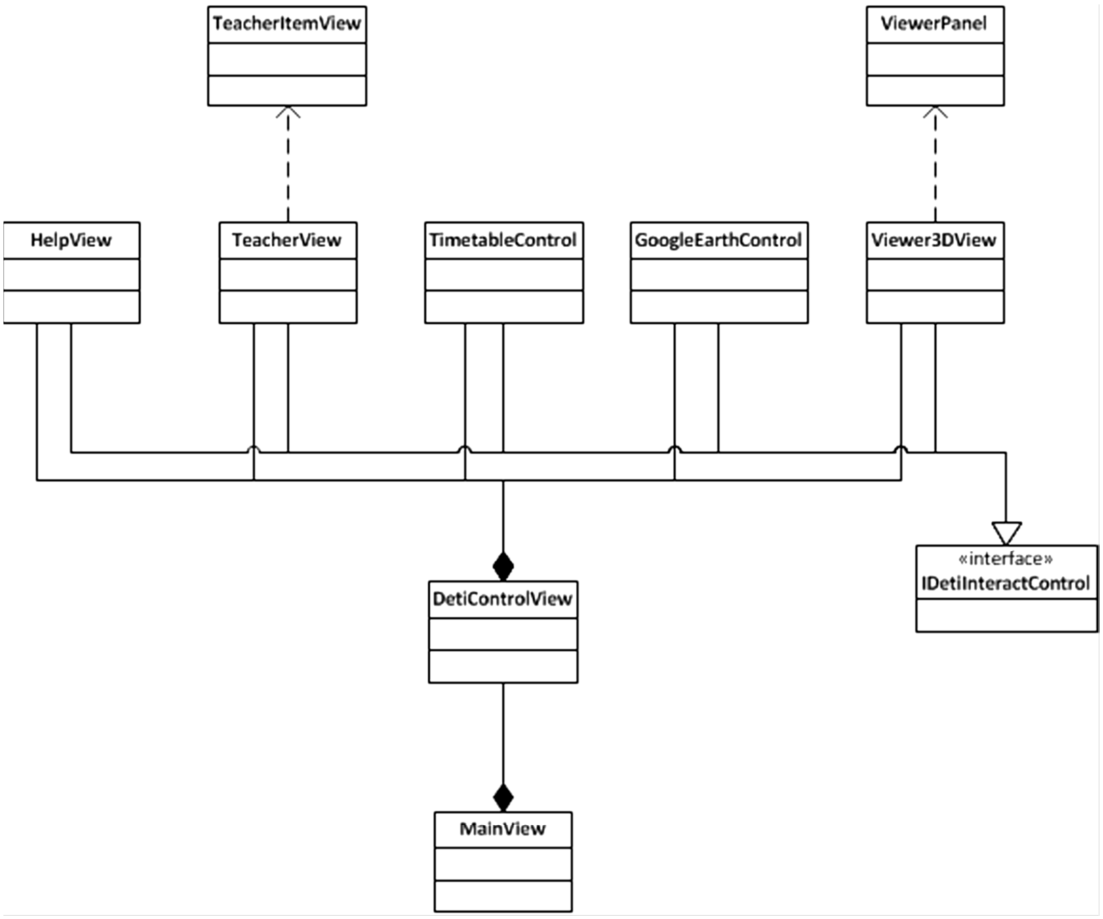

Figure 4.7 – Structure of the main application.

Each page must also display some content. In order to do so, the `IDetiInteractControl` interface is supplied. This interface provides methods that allow the system to perform operations such as triggering demonstration mode, and reacting to user input. Using these two components, all the developer has to do to add new content to the application is:

- Create a new WPF control to host the desired content.
- Implement the `IDetiInteractControl` interface. The control should have its own animation routine, and should respond to user input.
- On the main UI file (`MainViewModel`) used the provided method to add the control, and the title to be displayed on the page:
    - `AddWindowControl(new DetiControlView(new WPFControl(), "<Control Name>"));`

The developed control will have been added to the user interface, a new page will have been created, and the control will respond to user input and animation events.

4.3.2 Developed libraries

In order to offer the features discussed in Chapter 3, a set of auxiliary libraries was developed to provide the necessary content, namely, the DSD Provider library, and the Model Viewer XNA library. The DSD Provider communicates with the DSD server to collect the necessary information to present the lecturer list and the timetables, and the Model Viewer XNA library consists of an XNA application, which will render the models. Including a Google Earth plugin does not require the use of an auxiliary library, so it can be done within the main application.

In addition, two separate libraries were also developed, one that allows interaction from remote devices, the Control library, and one used to perform logging of events, the Logger library.

4.3.2.1 DSD Provider

As discussed in Chapter 3.2, DETI Interact needs data that is stored on the DSD platform, which is accessible via web services. Accessing web services, however, can take some time due to communication delays. Furthermore, processing the vast amount of data returned can also be time-consuming. These reasons prompted for the creation of the `DSDProvider` class library (Figure 4.8).

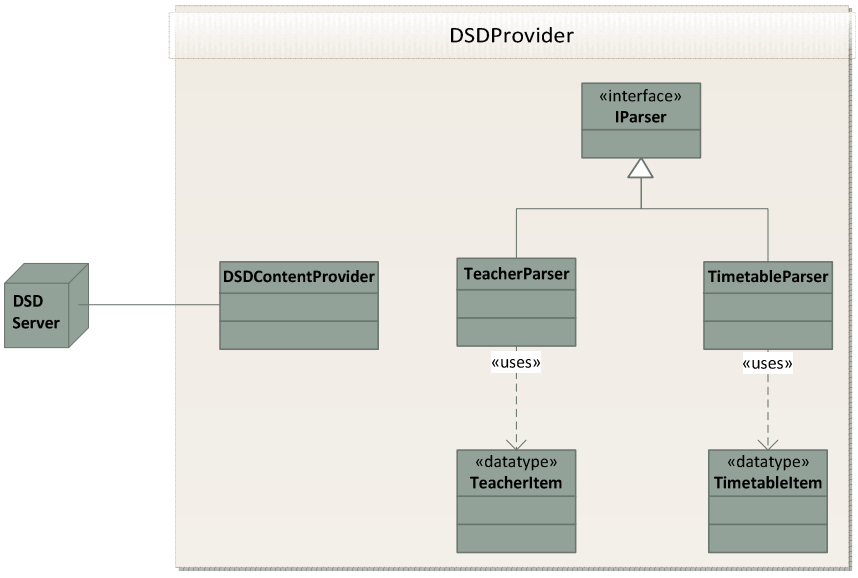

Figure 4.8 – The `DetiInteract.DSDProvider` class library.

This library is composed of two parts: the `DSDContentProvider` class, tasked with collecting all the data, and the Parsers, that work on the collected data.

4.3.2.1.1 DSDContentProvider

The `DSDContentProvider` class is responsible for accessing the Web Services and retrieving information from the server.

In order to use a Web service in a .NET project a proxy class must be created and compiled, this class is generated from a reference to the Web Service, typically the WSDL file. Using Visual Studio, this is achieved by adding a Service Reference to the project and assigning the address to the web services. Once the reference is added, the proxy class is generated automatically. The method for consuming a web service would be similar to that of calling a method from an external library.

4.3.2.1.2 Parsers

The Parser classes are functional components that work with the data provided by the `DSDContentProvider` class. Each parser processes a specific set of data and sends the result back to the application.

To ease system extensibility, an `IParser` interface was created. This interface lists the methods that every parser should implement. This allows the developer to determine how the parser will perform its functions and what data will be processed. The interface also publishes an event that can be used to notify the application of changes in the parser (such as when new data is available).

In the present state of the DETI Interact system, two distinct parsers are implemented: the `TeacherParser` and the `TimetableParser`.

In terms of performance, since calling Web services usually takes up more time than calling an internal operation, `BackgroundWorker` instances were created in each parser to collect and process the information. A `BackgroundWorker` allows an operation to be run on a separate, dedicated thread. Time-consuming operations which the parsers are likely to perform can make it seem as though the UI has stopped responding for a period of time, running it on a separate thread maintains responsiveness, improving user experience.

TeacherParser

The system that was previously running in the lobby contained a web page that displayed information regarding all the teachers of the department. The page would automatically scroll to different positions so that all would be shown. Information about each teacher was organized in a business card fashion with his or her name, photo, web page, office door number, email, and phone

number. This behaviour was to be reproduced in DETI Interact so this information had to be extracted from the list of teachers that the `DSDContentProvider` had fetched.

Extracting the required data from the `DSDContentProvider` is a simple operation although it requires looping through all the elements in the list. The `BackgroundWorker` loops through the list and extracts a teacher record, it then sends this record the `ProgressChanged` handler, which collects the required data and places it in a data structure. This structure is then sent to the UI thread via the existing event.

TimetableParser

The web page for the DSD platform (http://dsd.av.it.pt) allows the user to check timetables for each course and grade at the department, this feature was chosen to integrate the initial version of the DETI Interact system.

Creating a visual representation of a timetable requires a more careful approach since the Web services can only return textual information such as a list of all the subjects, classes, and classrooms, class times, etc., which constitutes an immense amount of data to be processed. In order to determine when a certain class takes place, it is necessary to navigate through six tables, which can be an arduous task for the CPU. Since the UI thread has to be able to quickly draw a timetable, most of the work will have to be done in the parser.

After accessing all the lists in the `DSDContentProvider`, the parser must extract the required data in order to create every item for the timetables. This data will be placed into a data structure (the `TimetableItem` class) that will be sent to the UI thread for drawing. The classic approach of looping through all the lists attempting to match IDs would result in an algorithm with a complexity of $O(N^6)$. This is where LINQ becomes a very useful asset. The querying capabilities of LINQ enable lists to be joined making it only necessary to loop through the resulting list once, lowering to complexity of the algorithm down to O(6N). This resulting list contains all the `TimetableItem` instances for a given year and course combination, and these can now be sent to the UI thread via the existing event.

4.3.2.2 XNA Model Viewer

One of the features previously discussed had the application display three-dimensional content. This would be used as a virtual tour of the department, allowing the user to visualize models or navigate in a virtual representation of the department.

There are numerous ways to display three-dimensional content on a computer, including OpenGL, DirectX, VTK, XNA, and WPF. Ideally, this 3D content would be rendered within the application instead of a standalone application. For this scenario, the first choice would be to use the 3D API included in WPF, as it would render content directly in the application window. However, this API is not as full-featured as OpenGL, DirectX, or XNA, meaning it would not scale as well with larger, more detailed scenes. The alternative APIs render 3D content to a specific buffer, such as the full screen buffer, or a given window handle, so it is required to replace this target buffer with a certain portion of the window.

Previously, rendering 3D content within an application required a Windows Forms control, which could be hosted within a WPF application. While this worked without problems, performance was severely affected. With version 3.5 of WPF the D3DImage `ImageSource` was introduced, which can be used as a render target for DirectX. Since XNA is a managed library built on top of DirectX, this was the library of choice for the Model Viewer.

The Model Viewer library is a standard XNA library project. It includes a series of 3D Models in FBX (Autodesk) or X (DirectX) formats, which are loaded and rendered by request of the UI thread. It also provides basic camera interaction.

4.3.2.3 Control

One of the main objectives of the DETI Interact system is the ability to control it via an Android mobile device. This is possible through the `DetiInteract`.Control library. This library is used to receive data from the Android device and interpret a certain gesture; it will then notify the UI thread of the gesture so that the application may perform accordingly. This implementation allows the application to decide how to deal with each gesture; it also allows the library to be reused in a different scenario.

This library contains a Controller class, and the `CommHandler` classes. In this version of the system, the chosen communication interface was Bluetooth, resulting in the creation of the `Bluetooth-`

`CommHandler`. Other `CommHandler` classes can be added as required, ensuring reusability and extensibility of the library.

4.3.2.3.1 BluetoothCommHandler

The `BluetoothCommHandler` class manages communication between Bluetooth enabled mobile devices. While the system is targeting Android devices in particular, any Bluetooth device will work with this class. All Bluetooth communication is handled by In The Hand's 32Feet.NET library (Foot). This library creates a higher level of abstraction on top of the Win32 API.

Setting up communication between Bluetooth devices is a usually straightforward task but Android's Bluetooth API requires those devices to be paired. Since the system is to be deployed to computers connected to a large display, with input methods inaccessible to the user, pairing must be done automatically and seamlessly. This presented a problem during development since different versions of Windows supported different Bluetooth specifications. Windows Vista and Windows 7 both support Bluetooth v2.1, which automatically performs Simple Secure Pairing with devices that support it. With Simple Secure Pairing, both devices agree on a sequence of characters, which the user then has to confirm on both devices. This feature overrode any attempt to pair the devices within the application, requiring user input on the Operating System level.

Windows XP did not present this problem, other solutions would include using and older Bluetooth adapter on the Windows machine, or using the Win32 API directly. The Android SDK version 2.3.3 features an API for insecure Bluetooth socket connections (CellPhonesMarket 2011), which will avoid this issue

To perform device pairing within the application a callback must be set to answer pairing requests. This callback only has to define the pin that the mobile devices must use to pair. This way pairing will be done and the Operating System will not interfere in the process.

Communication between devices using Bluetooth is similar to communication using a Serial interface (Bluetooth is essentially a wireless implementation of Serial communication). The devices must know the MAC address for each other, open a socket, and write into or read from that socket. Since the DETI Interact system will be deployed in a public setting, many Bluetooth-enabled devices will be in range, and polling every device would not be a very practical approach, instead the system will be constantly waiting for a device to request a connection. After accepting the connection, the system asks the device for the shared Stream. This stream will then be used to read the packets

sent from the mobile device that contain data pertaining to the gestures performed on the device. Since waiting for a connection request from a device is a blocking operation, and maintaining a continuous connection with a device is time consuming, the code must be run on a separate thread.

The data read from the stream encodes data from the mobile device. This data consists of a gesture identifier and a set of values corresponding to the gesture (for example, the X and Y speed of a fling); however it can also notify when a device is about to disconnect. This resets the listening process, leaving the system to wait for a new connection. Gestures are reported to the Controller via an event.

4.3.2.3.2 Controller

The `BluetoothCommHandler` class handles communication with remote devices via a Bluetooth connection and then report to the Controller class. Additional classes can be added to this library as needed to allow for different methods of communication. The `Controller` class' purpose is to manage the existing `CommHandler` classes. This class interprets that data received by the `CommHandlers` and notifies the UI thread via an event with details regarding to gestures performed on the mobile device.

4.3.3 Displayed Content

Various controls were included in the initial version of DETI Interact, namely a control that displays a list of teachers, a control that displays the students' timetables, a control that allows navigation in an interactive map, and a control that enables the user to manipulate a three-dimensional object. An additional page was added to the application displaying helpful information to the user regarding system usage.

As specified in the previous chapter, the displayed content will be organized in several pages, in a tabbed interface, with vertically oriented tabs arranged along the edges of the screen. Given the available set of gestures, it was decided that the touch-based gestures would be used to interact with the displayed content, while the orientation-based gestures would be uses to alternate between pages. The reason behind this choice was that the use of the digital compass would place the user under a higher cognitive load, so it would be more appropriate to leave it to secondary actions. This choice was later confirmed with user testing, which can be seen in Chapter 5. Interac-

tion with the content displayed in the pages is implemented by each page, which will be detailed below.

4.3.3.1 Teacher Page

As stated in 3.1, the previous system displayed a list of information regarding the teachers of the department. Information about each teacher was organized in a business card fashion and displayed in a matrix on a web page. This page would automatically scroll to pre-set locations so that all teachers would be displayed. The teachers' page in the new system implements a similar behaviour.

The teacher page consists of a Teacher Control, which hosts a list of `TeacherItemControls`. Both the controls are built using the MVVM pattern, taking advantage of the databinding capabilities of WPF. The `TeacherItemControl` was developed to house the information for each teacher. The ViewModel for the `TeacherControl` is responsible for dealing with the data. It starts by creating an instance of the `TeacherParser`, which returns a list of `TeacherItem` instances. These are used to create the `TeacherItemView` controls, which are placed into an `ObservableCollection`. WPFs databinding engine ensures that the contents of the listbox on the page match the contents of the collection at all times.

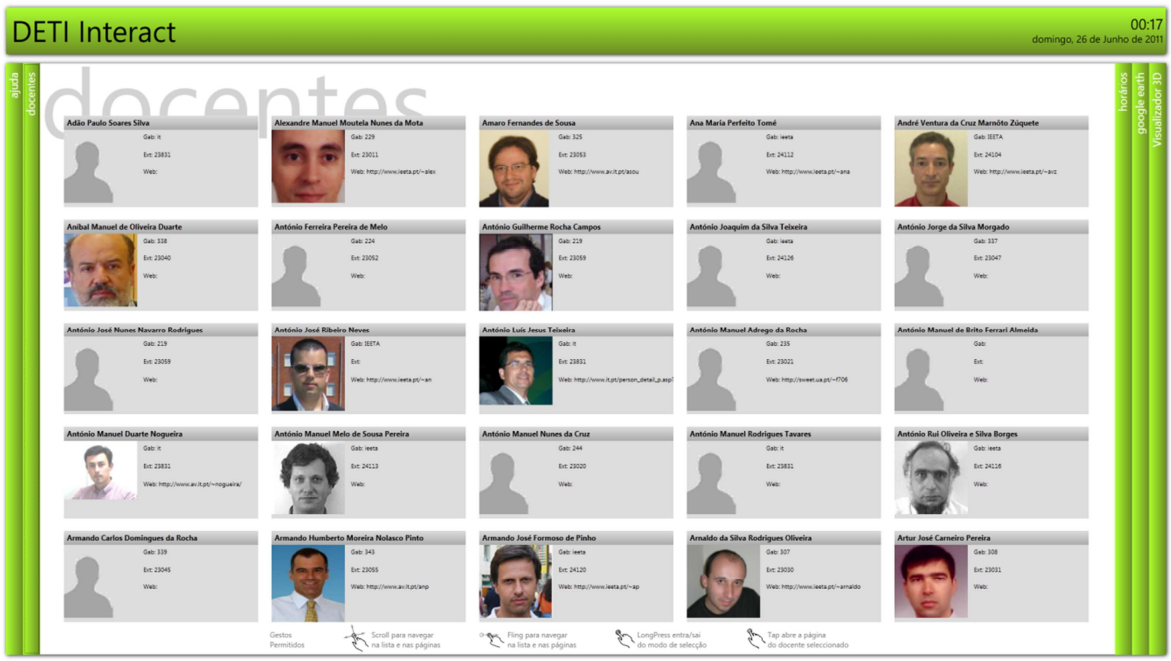

Figure 4.9 – The Teachers' page.

This control is configured to use all the available screen space, regardless of resolution, meaning that more information will be visible in the larger screens whilst remaining easy to read. For instance, a 1080p resolution screen will display five columns of teachers, while a 720p resolution screen will only display three columns. Figure 4.9 shows the final look for the teachers' page.

In addition to displaying the list of teachers, this control also displays the web page of any selected teacher if the URL is in the database. For the page to be displayed, a Web Browser control is overlaid on the list of teachers (Figure 4.10). When selecting a teacher, the address to his webpage is passed to the browser and the page is loaded. The Web Browser control was configured to prevent new windows from being opened, which would cause the application to lose focus, and become concealed by the newly opened window.

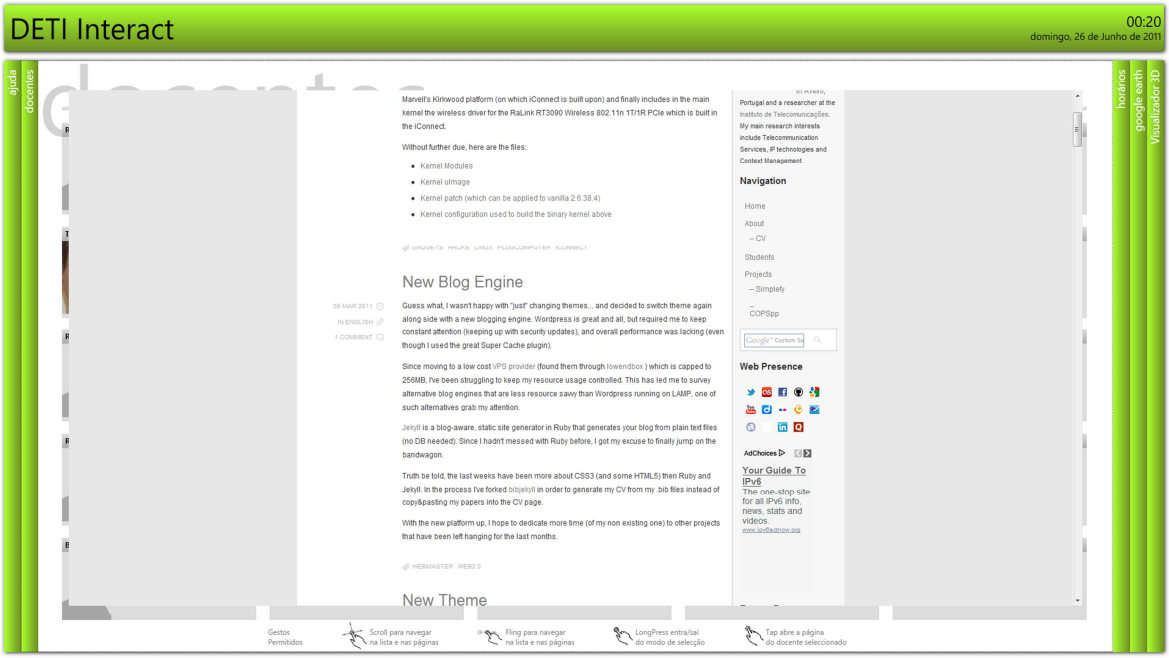

Figure 4.10 – The Web browser, visible after a user selects a teacher.

4.3.3.1.1 Interaction

To interact with this control, several gestures are provided. The users can scroll vertically along the list of teachers, or use a vertical fling gesture to trigger kinetic scrolling. In order to browse the teacher's webpage, the user must enter selection mode. This is done via a `LongPress` gesture. This gesture is used to enter and exit selection mode. Upon entering, the teacher closest to the centre of the page is automatically highlighted. The user can then use flings, in any direction, to select any

adjacent teacher. Once the desired teacher is selected, a simple Tap gesture will launch the web page.

Once the Web Browser control is visible and the webpage loaded, the user can scroll or fling, in any direction, to view the full extent of the webpage, in a manner similar to what is done in modern mobile web browsers. To leave the browser and return to the teacher list, the Long Press gesture is used once again.

4.3.3.2 Timetable Control

The previous chapter specified how the `TeacherParser` was used in creating the teachers' page; a similar approach was taken to develop the timetables' page, using the `TimetableParser`. The parser was built in a way that frees up the control of the greatest part of the work; however, the control still has to place every item in its correct place, in order to create a readable timetable.

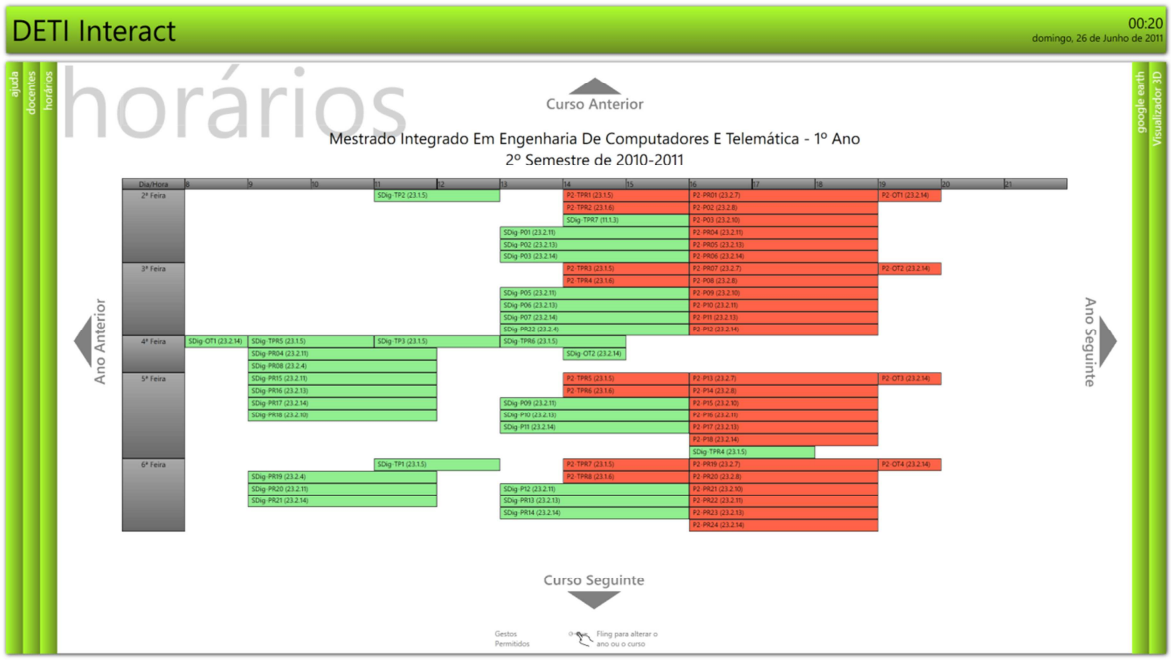

Figure 4.11 – The Timetables' page.

Analogous to the teacher control, an instance of the parser is created and every time the Changed event is triggered, the control draws to the screen. Data is passed to the control via a specific data structure containing subject name, classroom number, start time, duration, etc., called a `TimetableItem`. The timetable is represented by a Grid control where the rows represent days of the week, and the columns represent time. Before any work is done, the grid is configured for the appropriate degree program and year. The classes are then positioned on the grid, on their correct

places, colour-coded by course, so that they can be easily identified. In addition, since many classes can occur at the same, the grid's rows expand to accommodate them. The resulting timetable page can be seen in Figure 4.11.

4.3.3.2.1 Interaction

Interacting with the timetables via the Android device is limited to the Fling gesture. The user must swipe horizontally on the screen to navigate through the different years, or swipe vertically to change the degree program. Every time the user switches to a different timetable, the parser re-starts and process is repeated.

4.3.3.3 Google Earth Control

Another feature chosen to integrate the system is the inclusion of a Google Earth plugin in the application as it can also be used to validate the use of a mobile device for interaction with large displays, since it can offer a use for the touchscreen similar to what is already available on mobile devices and other touchscreen-equipped computers. Google Earth is presented on a page (Figure 4.12), and the user uses his/her phone to pan, tilt, or zoom.

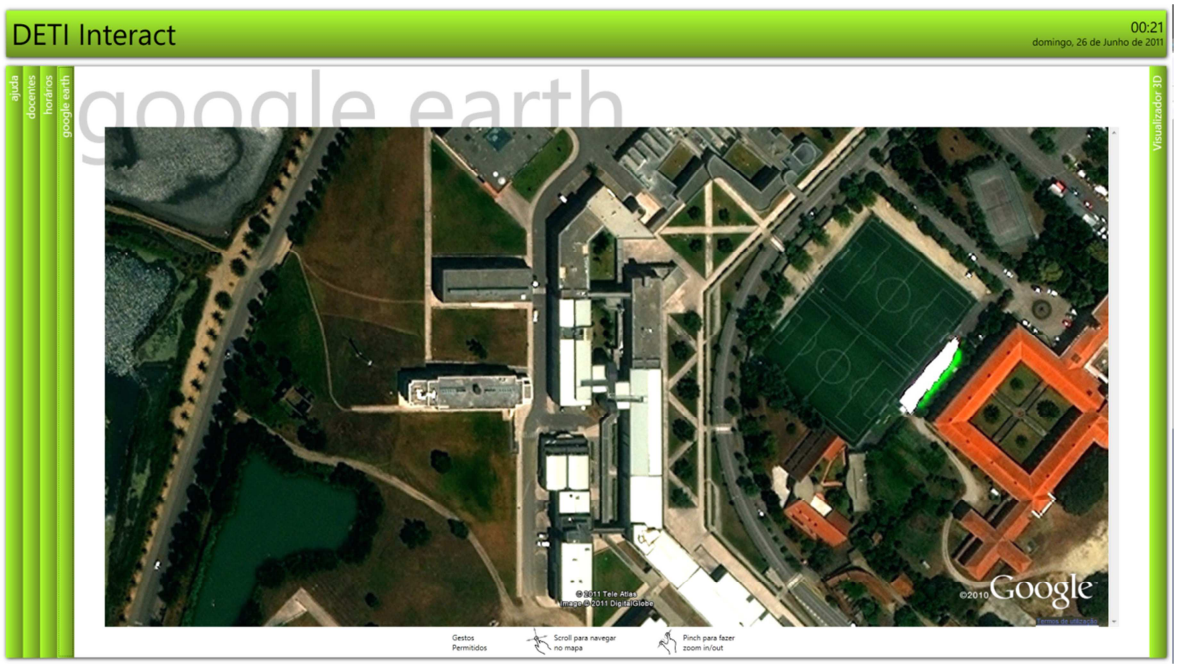

Figure 4.12 – The Google Earth page.

Google Earth works as a plugin integrated into a web browser. For developers, Google offers a Javascript API that allows web applications to host a Google Earth control. DETI Interact, however, is not a web application, meaning that there is no direct way to run the plugin in a native application.

In order to overcome this difficulty the developed control would have to be able to access the Javascript API from within the application. This was achieved in two steps. Firstly, it was necessary to find a way to draw a map on the screen, and allow interaction with it (panning, zooming, etc.). This was done via an HTML file. This file uses Google Earth's Javascript API to display the Google Earth control and works as a regular web page: it can be opened in a web browser, in which the Google Earth plugin is loaded and navigation within the plugin and interaction with it is as expected on any other page. The file also provides a small selection of Javascript functions that can be used to manipulate the map.

The second step consists in finding a way to render the HTML file inside the application, and using the provided functions to interact with the map. This is done by loading the file into a Web Browser control. This control renders the HTML content like any conventional web browser application, and using the provided `InvokeScript()` method, it is possible to call Javascript functions from within the HTML file. This resulted in the creation of new control to host the web browser

4.3.3.3.1 Interaction

Interaction with the map from the Android device is done as if the map had been drawn on the device's screen. Moving a finger in any direction will pan the map, and using the familiar pinch gesture, will zoom in or out, centred on the current position.

4.3.3.4 3D Viewer Control

The 3D Viewer control presents the content rendered by the XNA project mentioned in 4.3.2.2 within the WPF application. The 3D Viewer control is divided in two components: a panel to host the aforementioned D3DImage control and perform interoperation with XNA and DirectX, and a control to host this panel, which is tasked with animation and user interaction.

Viewer 3D Panel

The Viewer3DPanel hosts the D3DImage control and sets it up as the render target for the Model Viewer Library. This cannot be done directly although it is planned for future releases of both WPF and Silverlight (Microsoft 2011). This, however, can be achieved by using P/Invoke and Reflection to access specific DirectX and XNA content (Morril 2008).

In order to render a 3D scene into an image container inside the application, the render target of the XNA application must be switched to this container. This is done using the Interop Services

provided by the .NET platform. These services allow a developer to access features of native APIs, unavailable on the .NET framework, such as the Win32 API and DirectX. In the Viewer3D panel, P/Invoke is used to fetch the `IDirect3DTexture9` and IDirectDevice9 interfaces from the DirectX API. In addition, Reflection is used to get the private `ChangeDevice()` and `Initialize()` methods from the XNA API.

The Viewer3D Panel control can now access the `ModelViewer` library discussed in 4.3.2.2, and set the render target to the image container it hosts.

Viewer 3D

The Viewer3D is a control that wraps around the previously created panel. It was developed using the MVVM pattern. Its View Model is databound to the mentioned XNA application, while the view is responsible for animation and user input handling.

The Viewer3D control displays a list of all the three-dimensional models available in the system. It supports a selection mode similar to the one developed for the Teacher page. Once one of the models is selected, the Viewer3D Panel is displayed and interaction with the model is possible (Figure 4.13).

Figure 4.13 – The 3D Viewer Control, displaying a model synchronized with the mobile device's digital compass.

4.3.3.4.1 Interaction

Interaction with the Viewer page is similar to the interaction developed for the Teacher page. This page displays a list of the available models, in which the user can scroll of fling to browse its contents. A Long Press enters and exits a selection mode that allows the user to load a desired model via a Tap on the device's touchscreen. When a model is loaded, the Viewer 3D Panel is displayed, and the user can use his phone to manipulate the model. Orientation changes on the phone will directly affect the model, as if the user was holding the displayed model in his/her hands.

4.4 Deployment

In order to get the system working, the desktop application had to be installed and the mobile application had to be installed on the phone. The installation of the desktop application faced a series of problems. Since the application heavily depends on external sources (the Web Services from the DSD platform) and some features from the .Net framework, it was not possible to simply run the binary on the target machine. The application had to be installed in order for all the dependencies to be resolved. This was done using the provided ClickOnce functionality. This feature generates an executable binary for the application along with the needed assemblies and a configuration file. When the binary is run, the application is automatically installed and run.

Once the application was installed, it was also necessary to configure the machine so that it would run unattended. This involved setting up an automatic logon, configuring the anti-virus software and configuring the computer's software and services that run automatically during boot, so that DETI Interact can run without interruptions.

4.5 Conclusion

In Chapter 3, the system architecture was detailed, along with the two applications that would make up the system. The mobile device would work simply as a peripheral; therefore, the developed mobile application simply gathers data from the sensors and sends it via a Bluetooth connection. The desktop application then processes this data and interprets the gestures. The main function of the desktop application is to present content to the user. It connects to the required services and populates the screen. The system was developed and structured in a way that simplifies future modifications and extensions, so that additional content and interaction methods can be introduced to the system.

5 User Evaluation

In this chapter, the evaluation of the system will be discussed. In addition to user testing of the system, an application to evaluate Fitts' Law was developed to compare the use of a mouse versus the use of a mobile device as a means of interaction, in a series of selection tasks.

5.1 User Testing

An important stage in the development of an interactive system lies in user testing. Since the users will be the primary clients of the system, it must be ensured that they understand how to operate the system and enjoy using it. These tests assist in identifying errors in the system and problems in the user interface, which can then be addressed in order to deploy a fully functional version of the system.

DETI Interact proposes the use of a smartphone to control the display of information on a large screen. This was also studied during the development of DetiGuide (Palha 2010), the results, however, were inconclusive, consequently, with DETI Interact instead of proceeding straight to test the developed system, it was decided to conduct a preliminary testing session to evaluate Fitts' Law. The model developed by Paul Fitts relates the size of a target and its distance to the mouse cursor with the time it takes to select it. This would enable testing different usage scenarios where a smartphone could be used to control a computer. In addition to this test, a conventional test to evaluate the usability of the system was also organized.

5.1.1 Fitts' Law

In order to compare the ease of use of any given system using a mouse or a mobile device, an application was developed to evaluate Fitts' Law. Fitts proposed a formal relationship that models speed-accuracy trade-offs in rapid, aimed movement. This model predicts that the time required for a user to move to a target area is related to the distance and size of the target. It has been formulated mathematically in several different ways; the following equation, known as the Shannon formulation is preferred since it always provides a positive rating for the index of difficulty for each task (MacKenzie and Buxton 1992).

$$MT = a + b \log_2\left(\frac{2A}{W} + 1\right)$$

In the equation, **MT** represents the movement time to hit the given target, with **A** and **W** representing the distance to the target (amplitude), and the target's width, respectively. Constants **a**, and **b** are determined empirically with user testing. Once the results are processed, **a** expresses the start/stop time of the devices, and **b** represents the inherent speed of the device. The term $\mathbf{\log_2\left(\frac{2A}{W} + 1\right)}$ represents the index of difficulty, **ID**. In addition, $\mathbf{1/b}$ is called the index of performance, **IP**, of a device.

To compare the performance between the use of a mouse and mobile device, an additional application was developed, using .NET and WPF. This allowed for the reuse of components such as the Control library (described in chapter 4.3.2.3), which gives access to the Bluetooth stack as well as the gesture handlers. This application generates circular objects on de screen, which the user must click. The size and position of each circle on the screen is random, in order to randomize the index of difficulty. To ensure a wide range of data, a total of 50 targets are generated. There were five different circle sizes used on the application: 20px, 40px, 80pc, 120px, and 240px in diameter. The smaller sizes, 20px and 40px, were displayed with roughly 10mm and 20mm, which are confortable sizes to be used for finger-based interaction on a touchscreen (Microsoft 2011).

There were three different interaction methods available:

- The mouse, to be used as a reference for the other tests – the mouse is used to move the cursor, and the mouse buttons used to click on the targets;

- The touchscreen on the mobile device – the touchscreen is used to move the mouse cursor as a computer touchpad would, and tapping the touchscreen would send a click action to the application;

- The digital compass on the mobile device – device orientation manipulates the mouse cursor like a joystick, and tapping the screen clicks the targets.

Given the random nature of the tests, each user can perform the three tests. Since both mobile interaction methods are slower than the computer mouse, the mouse's sensitivity was reduced, to approximate the speed at which the others operate. All users were positioned at the same distance from the display, delimited by a table that would provide a surface for the mouse. Users remained standing throughout the tests.

5.1.1.1 Results

The tests were performed with 34 computer engineering students, resulting in 1700 results per test. All the tests were completed successfully, with the clear worst interaction being the use of the digital compass. The main problems with both methods of interaction on the mobile device were directly related to the accuracy in the collected values: the touchscreen has a much lower resolution than the computer mouse, and the digital compass, even in its fastest polling state, was too slow to manipulate the cursor comfortably.

In general, both methods have an overall worse performance when compared with the computer mouse. During the test, it was observed that when tapping the touchscreen users would often slightly move the cursor. This affected both interaction methods. The users felt that the smaller targets (roughly 1cm in diameter) were the hardest to click. In this case, both methods were distressing since very slight adjustments on the touchscreen were often detected as a Long Press by the Android OS, and the sensibility of the compass would have the users circling around each target without being able to position the cursor correctly. However, using the compass on these smaller targets still offered better precision. Apart from being more efficient with the touchscreen, some users enjoyed using the compass, since they could tap on the screen while tilting the device. In the end however, the preferred method of interaction was the touchscreen.

The results of this series of tests can be seen plotted in Figure 5.1. This graph shows the time it took for each user to select each target, using the three interaction methods. The horizontal axis expresses the index of difficulty, which relates the size of a target with the distance from the cursor.

The mouse interaction, used as a baseline, offers the best performance as it can be seen in the chart. Although the surface on which the mouse was placed was not optimal, which can be seen in some of the values, users were able to click the targets quickly. The interaction methods using the digital compass or the touchscreen on the mobile device show a worse performance, with the digital compass displaying the worst results.

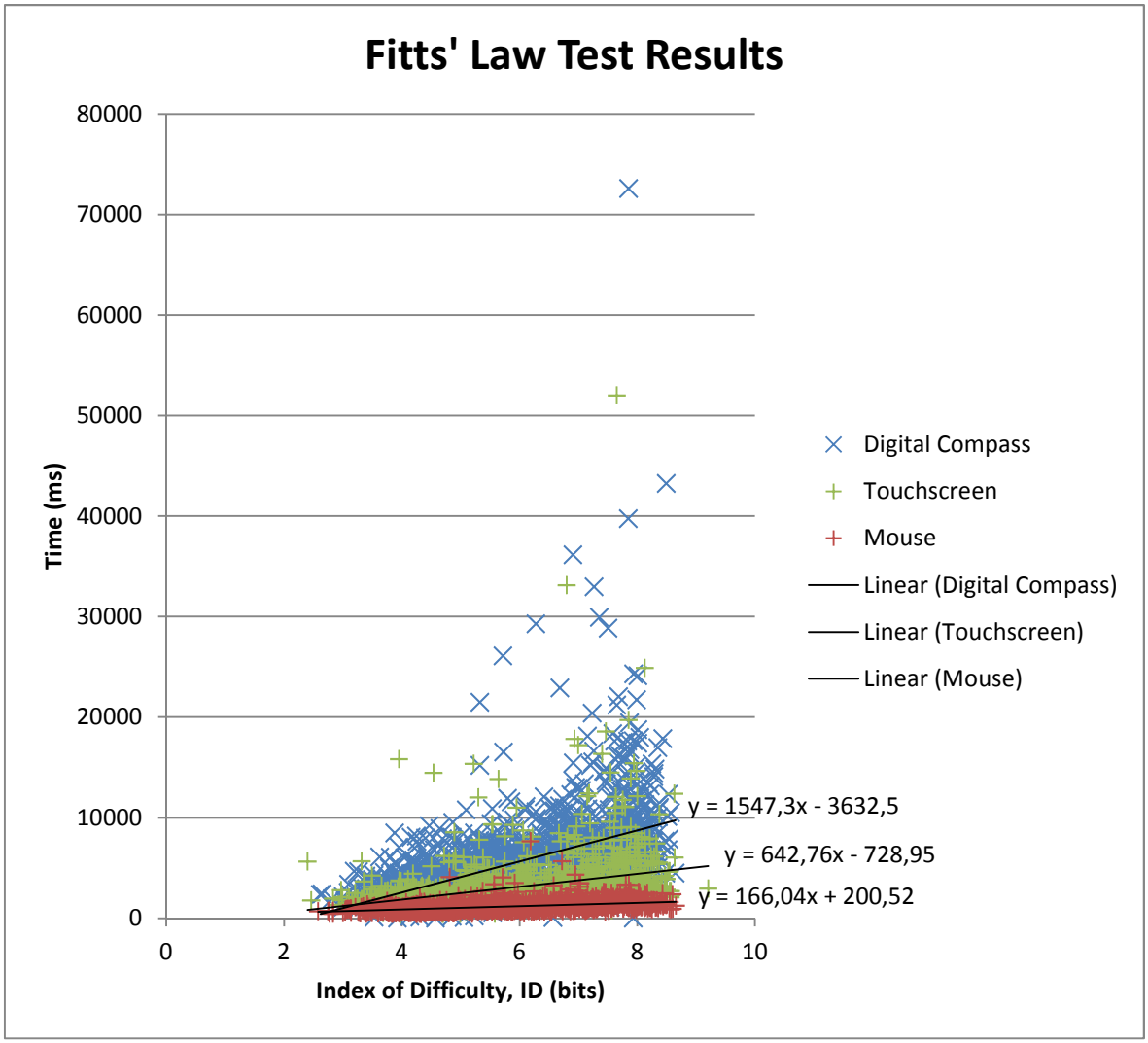

Figure 5.1 – Plot of the results obtained in the Fitts' Law test, along with a regression line for each method.

In addition to evaluating the time it took for the users to click on a target of a certain difficulty, it was also interesting to relate the elapsed time with the size of the targets. Figure 5.2 shows how long it took the users to click on each target, given its size. As specified previously, the size of the targets ranged from 20px to 240px. The graph shows that, as anticipated, users spent more time in attempts to click the smaller targets, with the larger targets earning the better results. This shows

that when creating user interfaces for this type of interaction, buttons, links, or other controls should have a considerable size, in order to provide the user with a better experience.

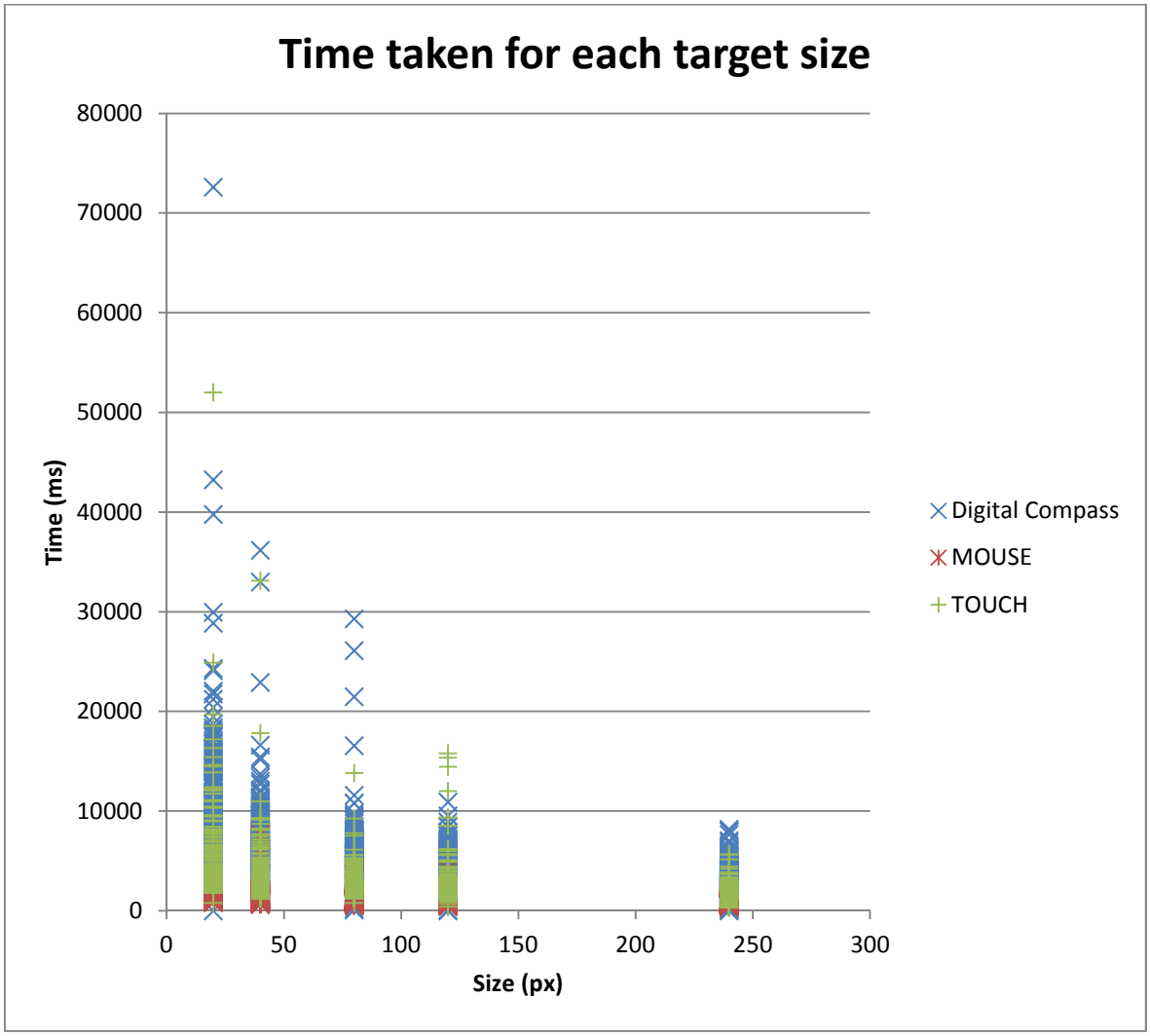

Figure 5.2 – Plot that relates the time taken to select each target with its size.

5.1.2 DETI Interact

The tests based on Fitts' Law assist in evaluating the touchscreen and digital compass of a mobile device as input methods, and validating their use in different scenarios. Nevertheless, interaction with the DETI Interact system must also be tested. Since many users are not familiar with modern smartphones, and given that touchscreen computers are more common in the scenario explored by this system, DETI Interact was tested on a computer with a touchscreen, as well as the large screens in the department's atrium. The touchscreen computer was not used during the Fitts' Law tests for two reasons: it had a much smaller resolution, which would affect the index of difficulty for each

target, and it allowed users to immediately tap the circle, which would evaluate his/her response time instead of the device's performance.

Interaction with the large screens was done with the smartphone, with the gestures discussed in chapter 4. Its touchscreen is used to control the content presented in each page, while the digital compass is used to alternate between the available pages.

In order to test the system, a list of tasks was prepared for the users to carry out. These tasks consisted of operations of varying difficulty, and ensured that the users would use all the provided features, simulating realistic usage of the system. Users were asked to complete these tasks and rate them regarding their difficulty. Once the tests were over, the users were asked to fill a small questionnaire where they could rate their experience and provide some comments about the system. The proposed tasks were the following:

1. Connect to the system;
2. Find a phone extension for a certain teacher;
3. Find more detailed data for another teacher by accessing his or her website;
4. Check the classroom for a given course;
5. Navigate to a location on Google Earth;
6. Manipulate a three-dimensional model.

The tasks were the same on both computers; however, the Google Earth plugin was not operating correctly on the touchscreen computer, which resulted in the termination of the application. This led to the creation of two different versions of the application, a fully featured version, running on the large screen, and a version without the Google Earth plugin, running on the touchscreen computer. Users on the touchscreen were asked to skip the task referring to navigation using Google Earth.

5.1.2.1 Results

This test was performed with 22 computers engineering students. All the tasks were completed successfully, with both devices showing favourable results. Figure 5.3 shows the average results for the difficulty of each task, with one being very hard, and five being very easy. Apart from the tasks

that the touchscreen could not run – connecting via the mobile application and using Google Earth – it can be seen that the scores between both computers are very similar.

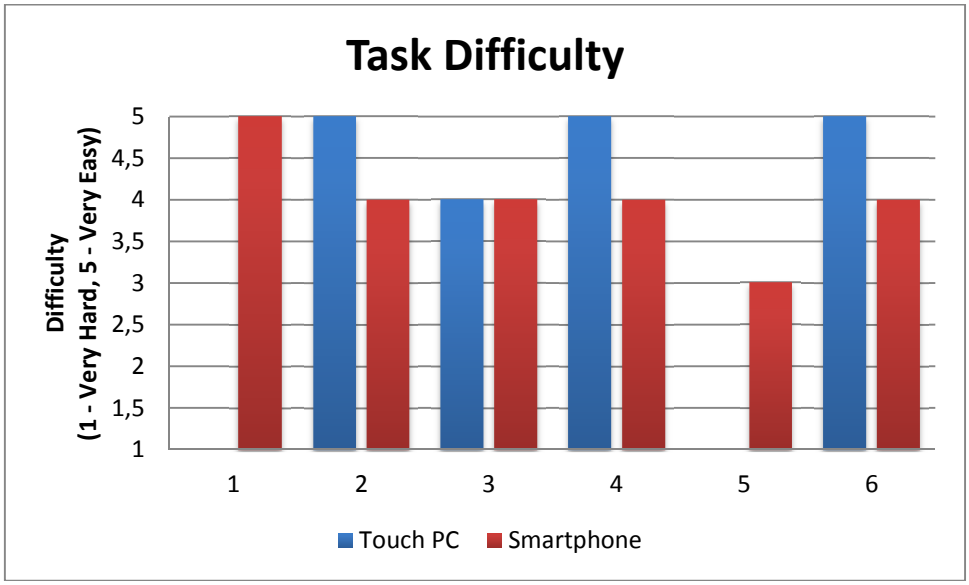

Figure 5.3 – Difficulty of the tasks, expressed by the users, comparing the two interaction methods.

When users connect to the system, a help page is automatically displayed and, in addition, assistance on how to interact with each page is provided at the bottom of the pages. These measures ensure that the users do not feel lost or confused when operating the system. During the observation of the tests, the appropriate gestures were performed. One of the main complications with the mobile device was related with the direction of movement. Since not many users were familiar with modern interaction methods, such as scrolling a list of items on a mobile device, users sometimes expected the system to respond differently. For instance, scrolling down the list of teachers is done by moving the finger upwards on the touchscreen. This simulates the action of dragging the list upwards, in order to reveal the content below. This type of gestures is currently used on most touch-based mobile devices, which led to them being implemented this way on the system. Additionally, users also made mistakes when alternating between pages, which is done by tilting the device sideways. They would either tilt the device in the wrong direction, or attempt to change pages with a horizontal swipe on the touchscreen. Another complication common with most users was interacting with Google Earth. As explained in section 4.2, the pinch gesture was unavailable in the used version of the API, which led to this feature being implemented from scratch. The resulting gesture was very sensitive which left users sometimes feeling out of control.

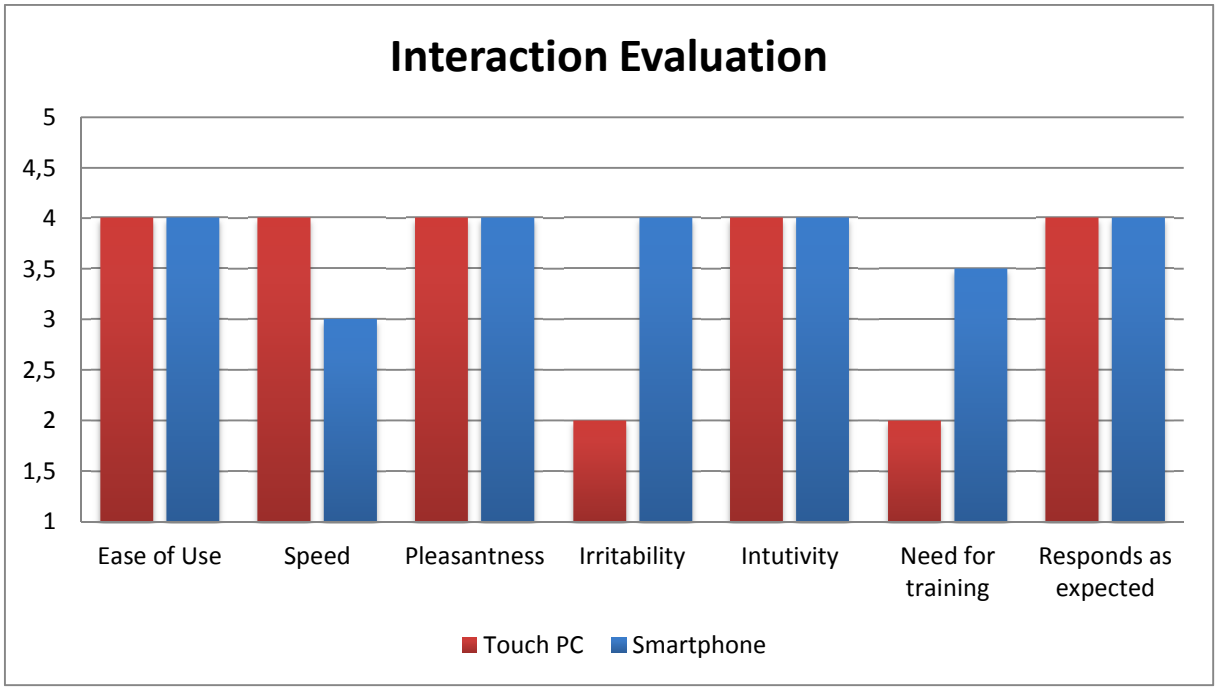

Figure 5.4 - Results of the queries answered by the users after the completion of the tasks.

Following the testing session, users were asked to fill a small survey regarding their experience when using the system. In this survey, they would rate the system concerning ease of use, speed, pleasantness, irritability, intuitiveness, need for training, and responsiveness. The results indicate that the overall experience using the smartphone was inferior when compared to the touchscreen (Figure 5.4). While most users felt that the system was intuitive and easy to use, the lowest scores lie in the speed, irritability and need for training. This, however, was not unexpected. The overall speed of the system was affected by the amount of data being drawn to the screen. As explained on section 3.4.2, the computer running the software presented low performance. With the list of teachers consuming a lot of memory, and the 3D Model Viewer's use of the GPU, which was aggravated by the use of reflection in the code, the system suffered severe hits on performance. In addition, as explained above, some users tried tilting the device, or scrolling in the wrong direction, and most of the users where overwhelmed by the sensitivity of the pinch gesture in the Google Maps page.

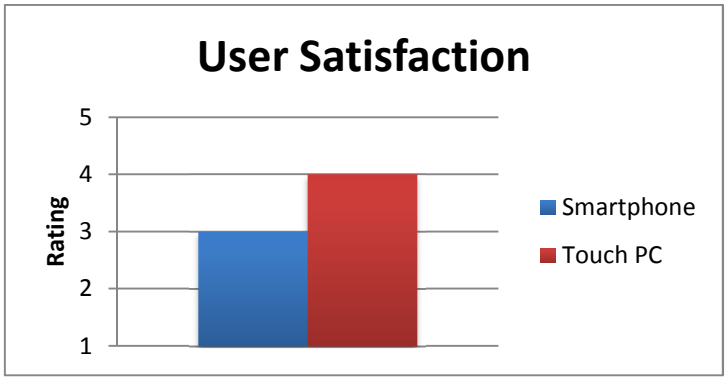

Figure 5.5 – Satisfaction of the users when using either interaction method.

Users were also asked to rate their overall satisfaction with the system (Figure 5.5). Once again, the use of the smartphone received worse ratings. When asked for their preference, users also selected the touchscreen-equipped computer as their favourite, although some users did choose the smartphone over the touch PC (Figure 5.6).

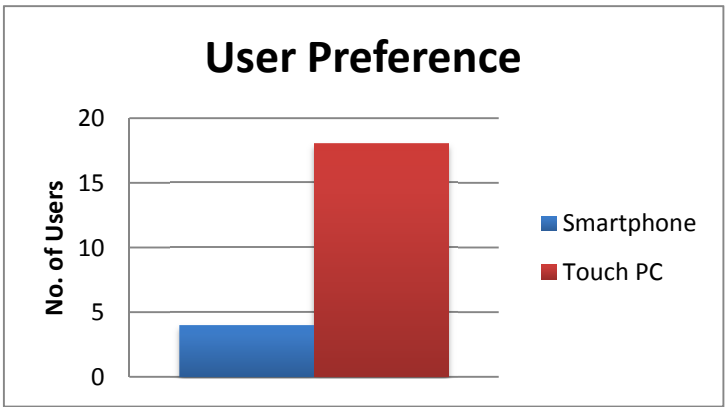

Figure 5.6 – User preference on the interaction method.

In the survey, users had the opportunity to write some comments and give suggestions on how to improve the system. Almost all the comments criticized the sensitivity of the zoom gesture on Google Maps. Some suggested that a double tap gesture could be added to the system to complement the zooming gesture. There were also some references to the titles of the pages, displayed vertically on the sides of the pages, had a very small font. When asked to suggest new features, nearly all the participants requested that the menus for the canteens be displayed on the application. In addition, some news relative to the department, as well as the university, could also be displayed.

5.2 Discussion

Testing the system with users is an important stage when developing software. The objective is to identify errors in the system and determine areas where improvements can be made. For this project, two different testing waves were organized. The first one addressed Fitts' law and its implications. The second involved testing the developed system. Testing was always done with different interaction methods, in order to compare the system with other real world scenarios.

As explained previously, Fitts' Law establishes a relation between the size of a target and its distance from the cursor with the time it takes a user to select it. Data gathered from this test can be used to design better user interfaces. Since DETI Interact proposes the use of a smartphone to control information displayed on a large screen, this test compared the use of the touchscreen and the digital compass with the use of the computer mouse. Both of the studied methods are naturally more arduous when compared with the mouse, but between them, the use of the digital compass offers the worst control method. From the results, it can be concluded that the digital compass should not be used for tasks that require some dexterity. The touchscreen, on the other hand, fares well in these situations.

Considering these results, the interaction methods chosen for DETI Interact – using the touchscreen for most of the operations, and the digital compass to alternate between pages – where considered valid, although some adjustments or alternative gestures could also be provided. In addition, the results also indicate that in scenarios where the user might need to control a cursor or select certain objects, these must have a practical size, with at least 100 pixels in diameter. Furthermore, alternative methods of interaction might be considered. These methods might include different speed or acceleration for the cursor movement, depending on the distance to the target, or using an approach where the targets generate gravity, which would assist the user in the selection.

After processing all the data from the first wave of tests, users were asked to participate in a second wave, in order to evaluate the usability of DETI Interact. While this second wave of tests did not provide the most favourable results towards the system, these were close to those obtained from similar devices (the touchscreen-equipped computer). Considering the results gathered from the previous testing session, concerning Fitts' Law, and the observed reactions of the users, the chosen library of gestures was deemed fitting. While the learning curve for the system may be steeper when compared with using direct interaction methods, users still felt that the system was easy to use, albeit with some initial guidance.

Some users complained about the slow performance of the system. As explained above, this had to do with the implemented 3D Model Viewer, alongside the low performance of the system. This feature was used as a means of comparing three-dimensional interaction techniques using a mobile device equipped with a digital compass, accelerometer, or gyroscope, with direct methods of interaction, such as the touchscreen computer. Given its utility for the department, and the limited availability of 3D models, this feature was removed from the version of the system currently running in the computers of the department. The amount of data displayed on the teachers' page also hinders system performance, although the hit is less severe. Another complaint was concerning the sensitivity of the pinch to zoom gesture on the mobile device. The ideal solution for this problem would be to use the API for Android 2.2, which includes the appropriate functions.

Since the testing, some changes were made to the application, mostly to the user interface such as, some font sizes were increased to improve readability, and the titles of the pages were added to the background of the pages to better situate the users. Visual help was also improved.

6 Discussion

6.1 Conclusion

The main purpose behind DETI Interact was to develop a system that displayed information concerning the Department of Electronics, Telecommunications, and Informatics of the University of Aveiro, and allowed users to interact with this information via an Android smartphone. This would allow users to interact with a system that did not support any type of direct or indirect interaction, such as touchscreens. The system consisted of an application that ran on a computer connected to a large display, which is located in the lobby of the department. An application for Android based devices was also developed which was responsible for the communication between the mobile device and the computer.

6.1.1 Development

As described in chapters 3 and 4, the system communicates with the DSD platform, the department's server that holds information regarding lecturers, students, classes, etc., to gather data relating to the department. It also uses the Google Earth API to display a map plugin and the XNA framework to display and allow interaction with three-dimensional objects.

The desktop application was developed using .NET and WPF. These technologies were helpful in prototyping and developing the system rapidly, as well as offering a greater performance. Given the low hardware specifications of the target machines, it might have been worthy to attempt to develop the system in a lower level language (using C++ with QT for the user interface, for example). An

attempt was made using Windows Forms, but the large amount of data drawn on screen was significantly worse, when compared to WPF. Using WPF was also helpful in developing a system that was easily modifiable and extensible, for its use of the MVVM pattern.

The Android mobile application was developed in Java. While Android offers a good development environment when compared to other mobile platforms, its programming model, using the MVC Pattern, and the provided API, feels cluttered, which degraded the developing experience. The main problems were encountered with the Bluetooth communication. It was chosen for its simplicity and lack of configuration needed by the user, but the API provided in the Android SDK damaged the experience that could be provided to the users of DETI Interact.

Overall, the development of DETI Interact did not face any major setbacks, due to well-defined system requirements and suitable choice on the development platforms.

6.1.2 Results

The system was deployed to the lobby where it has been running since January 2011. During this time, it has been receiving updates and generating usage logs, which assisted in its development and maintenance.

The system was later tested with users to evaluate the performance of the smartphone as an input peripheral, as well as the usability of the application. The first testing wave targeted the evaluation of the performance of the smartphone. It was based upon Fitts' Law and yielded interesting results, which are described in detail in Chapter 5. The results suggested that the use of the touchscreen of a mobile device could be an appropriate input peripheral, while the use of the digital compass could be more problematic. This corroborated the choices made during development, where the compass was used for switching pages, by tilting the device sideways, and the touchscreen was used to interact with the content presented on each page.

A second wave of testing evaluated the usability of DETI Interact. Users were given a series of tasks to fulfil and then rate their difficulty. In the end, they also participated on a small survey where they rated their experience and commented the system. The results were compared with the use of a touchscreen-equipped computer, which would be a more familiar device. While the touchscreen computer received better results, these remained close to the results obtained from the smartphone. Users thought that, while the system was easy to use, some usage scenarios were

more irritating in the smartphone, and that the learning curve was steeper. While at first glance the results seem unfavourable towards the smartphone, it must be taken into account that these were compared with other already well-established methods of interaction, such as the computer mouse, with which users are more experienced. However, the mouse itself took thirty years to become a ubiquitous device (Buxton 2007). Natural User Interfaces (NUI), such as those explored by voice, gesture, and touch, albeit quickly becoming embraced by the industry and the public, are still struggling to become ubiquitous. They are still criticized for not being as good as a computer mouse for most tasks, even though they might be superior in other tasks. As William Buxton says: "everything is best for something and worst for something else", also adding, "the trick is knowing what is what, for what, when, for whom, where, and most importantly, why. Those who try the replace the mouse play a fool's game. The mouse is great for many things. Just not everything" (Buxton 2007). It was not unexpected that the interaction performance with the mobile device would be worse, but it allows for a good method of interaction in a situation where other alternatives would not be viable or would require a greater investment.

In conclusion, DETI Interact proposed to study approaches to interaction with large displays, and implement and information system on the lobby of the Department of Electronics, Telecommunications, and Informatics of the University of Aveiro. Although the results could have been superior, the system received good reviews, and is currently running on a on a computer in the department. While direct manipulation of data on a remote screen via a smartphone might appear to be an arduous task for users, given the impossibility of using a computer mouse, an appropriate user interface and set of gestures can make a system usable, with a promising user experience.

6.2 Future Work

As with any software project, there are always areas that can be improved, and features that can be added. The results and comments gathered from the testing sessions were invaluable in finding areas in need of improvement. The principal complaint about the system was in regards to the zoom gesture in the Android device. This had to do with the gesture not being available in the API and having to be written from scratch. While the sensibility can be improved, version 2.2 of the API includes the classes that handle this gesture, which would provide the best user experience.

In addition, alternative gestures can be implemented, such as the Double Tap, also a common in zooming tasks. Furthermore, the system could use and extended gesture library. For instance, some

users would try to use the Fling gesture to alternate between pages, instead of tilting the device. Since the touchscreen allows detection of multiple points, a two-finger swipe can be implemented.

In respect to new features, users suggested that the system could also display the menus for the canteens, ticket information from the university's administration services, as well as news from the department. Additional information such as exam dates, vacant classrooms, integration with the university's library, was also suggested by other students.

Apart from the functionalities suggested by the users, other features were already being considered. One interesting feature would be to make the system context-aware. Once a user connected to the system, he would be identified and specific information could be displayed, such as his timetable, exam dates, project deliveries, etc. This would require further integration with the university's systems such as PACO (*Portal Académico Online*) and Moodle. Additional information could also be displayed on the mobile device's screen. The mobile device is currently used solely as an input peripheral, but the available screen can also be used for information display. As stated in chapter 4, Android 2.3 allows for the use of Insecure Bluetooth Sockets. This removes the need for the devices to pair, which would improve the user experience on the first connection to the system. However, using a more recent version of Android (whether it is 2.2, for the scale gesture, or 2.3, for the insecure Bluetooth sockets) will exclude users running older versions.

In addition, other mobile platforms could also be included. While not all provide access to the Bluetooth stack, future updates could bring such functionality, and other communication methods are available (e.g. WiFi, Sockets). Furthermore, given the results obtained from the first testing session, other interaction methods could be added to the system, such as the ability to control the mouse cursor. This would allow users to navigate webpages, for example, as opposed to simply viewing them as is currently done. It could also improve selection-based tasks, such as those existent in the teacher page. DETI interact was developed in a way that interaction with the system is not dependent on any device, which allows for the system to support additional devices such as video cameras or the Microsoft Kinect device. The system could also be extended in order to allow multiple users in a collaborative environment, as well as the possibility of spanning across additional displays. Augmented reality is also an option that can enrich the experience of the user. Testing also revealed that the mobile device offers a suitable method of interaction for 3D content, meaning that further applications of three-dimensional content should be studied.

References

Anthony, S. (2010). "Microsoft Kinect controlling Windows 7, exciting proof of concept." Retrieved 25/11/2010, from http://downloadsquad.switched.com/2010/11/22/microsoft-kinect-controlling-windows-7-exciting-proof-of-concep/.

Boring, S., D. Baur, et al. (2009). "Touch Projector: Mobile Interaction through video." Proceedings of CHI 2010, Atlanta, GA, April 10-15: 2287-2296.

Buxton, W. (2007, 21/03/2011). "Multi-Touch Systems that I Have Known and Loved." Retrieved 18/06/2011, from http://www.billbuxton.com/multitouchOverview.html.

Campos, D. (2008). Distribuição de Serviço Docente: back e front office Web. Departamento de Electrónica, Telecomunicações e Informática, Universidade de Aveiro. **MSc:** 148.

CellPhonesMarket. (2011). "Google's new SDK features an API for "insecure Bluetooth socket connections"." Retrieved 21/05/2011, from http://www.cellphonesmarket.com/news/googles-sdk-features-api-insecure-bluetooth-socket-connections/.

Cheng, K. and K. Pulo (2003). Direct interaction with large-scale display systems using infrared laser tracking devices. Proceedings of the Asia-Pacific symposium on Information visualisation - Volume 24. Adelaide, Australia, Australian Computer Society, Inc.**:** 67-74.

Cull, T. (2010). "Researchers Highlight Recent Uptick in Java Security Exploits." Retrieved 14/02/2011, from http://www.infoq.com/news/2010/10/java-exploit-uptick.

DailyTech. (2011). "Android Overtakes Symbian as World's No. 1 Smartphone Platform." Retrieved 14/02/2011, from http://www.dailytech.com/Android+Overtakes+Symbian+as+Worlds+No+1+Smartphone+Platform/article20787.htm.

Finke, M., A. Tang, et al. (2008). Lessons learned: game design for large public displays. Proceedings of the 3rd international conference on Digital Interactive Media in Entertainment and Arts. Athens, Greece, ACM**:** 26-33.

Fling, B. (2009). Mobile Design and Development, O'Reilly.

Foot, P. "In The Hand .NET components for mobility." Retrieved 04/11/2010, from http://inthehand.com/.

GIZMODO. "Orange Shows Off Gesture Based Interaction Screen, Touch Screens Look On in Horror." Retrieved 04/11/2010, from http://gizmodo.com/346845/orange-shows-off-gesture-based-interaction-screen-touch-screens-look-on-in-horror.

Google. (2005). "Google Earth." Retrieved 13/06/2011, from http://www.google.com/earth/index.html.

Google. (2010). "Features - Google TV." Retrieved 24/11/2010, from http://www.google.com/tv/.

Google. (2011). "What is Android? | Android Developers." Retrieved 12/04/2011, from http://developer.android.com/guide/basics/what-is-android.html.

Hardy, R. and E. Rukzio (2008). Touch \& interact: touch-based interaction of mobile phones with displays. Proceedings of the 10th international conference on Human computer interaction with mobile devices and services. Amsterdam, The Netherlands, ACM**:** 245-254.

Hardy, R. and E. Rukzio (2008). Touch \& Interact: touch-based interaction with a tourist application. Proceedings of the 10th international conference on Human computer interaction with mobile devices and services. Amsterdam, The Netherlands, ACM**:** 531-534.

Ideum. (2011). "Supported Multitouch Gestures." Retrieved 16/06/2011, from http://gestureworks.com/support/supported-gestures/.

Kaviani, N., M. Finke, et al. (2009). "Encouraging Crowd Interaction with Large Displays using Handheld Devices " Proceedings of CHI 2009, Boston, MA, April 4-19.

MacKenzie, I. S. and W. Buxton (1992). Extending Fitts' law to two-dimensional tasks. Proceedings of the SIGCHI conference on Human factors in computing systems. Monterey, California, United States, ACM**:** 219-226.

Madhavapeddy, A., D. Scott, et al. (2004). "Using Camera-Phones to Enhance Human-Computer Interaction." Proceedings of Ubiquitous Computing (Adjunct Proceedings: Demos): 1-2.

Microsoft. (2009). "Microsoft .NET Framework." Retrieved 13/06/2011, from http://www.microsoft.com/net/.

Microsoft. (2010). "Introduction to WPF." Retrieved 20/08/2010, from http://msdn.microsoft.com/en-us/library/aa970268.aspx.

Microsoft. (2010). "Kinect - Xbox.com." Retrieved 25/11/2010, from http://www.xbox.com/en-US/kinect.

Microsoft. (2011). "Kinect for Windows SDK from Microsoft Research." Retrieved 16/06/2011, from http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/.

Microsoft. (2011). "Microsoft Expression Blend® 4 | Silverlight | Rich Internet Applications | XAML | WPF Applications | .NET Platform | TFS | VB | C# | Microsoft® Expression®." Retrieved 13/04/2011, from http://www.microsoft.com/expression/products/Blend_WhatIsExpressionBlend.aspx.

Microsoft. (2011). "Silverlight 5 beta: The Official Microsoft Silverlight Site." Retrieved 13/06/2011, from http://www.silverlight.net/getstarted/silverlight-5-beta/.

Microsoft. (2011). "Touch." Retrieved 25/05/2011, from http://msdn.microsoft.com/en-us/library/cc872774.aspx.

Mihailescu, D. (2010). "Benchmark start-up and system performance for .Net, Mono, Java, C++ and their respective UI." Retrieved 14/02/2011, from http://www.codeproject.com/KB/dotnet/RuntimePerformance.aspx.

Miyaoku, K., S. Higashino, et al. (2004). C-blink: a hue-difference-based light signal marker for large screen interaction via any mobile terminal. Proceedings of the 17th annual ACM symposium on User interface software and technology. Santa Fe, NM, USA, ACM**:** 147-156.

Morril, J. (2008). "XNA, Meet WPF." Retrieved 04/11/2010, from http://jmorrill.hjtcentral.com/Home/tabid/428/EntryId/259/XNA-Meet-WPF.aspx.

Palha, R. (2010). Interacção entre um dispositivo móvel e um ecrã de grandes dimensões. Departamento de Electrónica, Telecomunicações e Informática, Universidade de Aveiro. **MSc**.

PhoneGap. (2008). "PhoneGap." Retrieved 11/04/2011, from http://www.phonegap.com/.

Richter, J. (2010). CLR via C#, Microsoft Press.

Samsung. (2010). "Samsung Internet @ TV." Retrieved 24/11/2010, from http://www.samsung.com/uk/internet-tv/.

Seewoonauth, K., E. Rukzio, et al. (2009). Touch & connect and touch & select: interacting with a computer by touching it with a mobile phone. Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services. Bonn, Germany, ACM**:** 1-9.

Stogaitis, M. and M. Sun. (2008). "Gmote - Android Remote." Retrieved 21/11/2010, from http://www.gmote.org/.

Tang, A., M. Finke, et al. (2008). Designing for bystanders: reflections on building a public digital forum. Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems. Florence, Italy, ACM**:** 879-882.

TheAlternative. "The Alternative Homepage." Retrieved 21/11/2010, from http://www.thealternative.co.uk/.

Thurrot, P. (2010). "More work on Chapter 4 and notes about the origins of Metro." Retrieved 14/05/2011, from http://windowsphonesecrets.com/2010/04/04/more-work-on-chapter-4-and-notes-about-the-origins-of-metro/.

Thurrott, P. (2010). "Xbox 360 Dashboard Screenshots Gallery 1." Retrieved 14/05/2011, from http://www.winsupersite.com/article/product-review/xbox-360-dashboard-screenshot-gallery-1.

Totilo, S. (2010). "Natal Regonizes 31 Body Parts, Uses Tenth of Xbox 360 "Computing Resources"." Retrieved 01/03/2011, from http://kotaku.com/5442775/natal-recognizes-31-body-parts-uses-tenth-of-xbox-360-computing-resources.

Underkoffler, J. (2010). "John Underkoffler points to the future of UI." Retrieved 01/03/11, from http://www.ted.com/talks/john_underkoffler_drive_3d_data_with_a_gesture.html.

Vogel, D. and R. Balakrishnan (2004). Interactive public ambient displays: transitioning from implicit to explicit, public to personal, interaction with multiple users. Proceedings of the 17th annual ACM symposium on User interface software and technology. Santa Fe, NM, USA, ACM**:** 137-146.

W3C. (2004). "Web Service Glossary." Retrieved 04/11/2010, from http://www.w3.org/TR/ws-gloss/.

Wired. (2009). "5 Nifty Apps That Turn Your Android Into A Universal Remote." Retrieved 21/05/2011, from http://www.wired.com/gadgetlab/2009/11/5-nifty-remote-apps/.

Appendix I – User Tasks and Questionnaire

After the tests conducted on DETI Interact, users were asked to fill the following questionnaire:

DETI Interact: Questionário do Utilizador

Instruções: Agradecemos a sua colaboração na realização deste estudo, que tem por objectivo avaliar a *Interface de Utilizador* do *DETI Interact* e, consequentemente, tentar melhorá-la seguindo os critérios de *Usabilidade*.
A sua colaboração constitui um factor importante para o êxito desta avaliação, por isso solicitamos-lhe o preenchimento deste questionário, cujos dados serão usados com total anonimato apenas para fins científicos.

1. Dados pessoais

Nº Mecanográfico: _____ Turma:_____ Data:_____/_____/_____
Género (M/F): _____ Idade: _____ Nº de Utilizador: _____

Tem experiência com dispositivos que possuem ecrã sensível ao toque, acelerómetro ou bússola digital (por exemplo *smartphones*)? S___ N___

Se respondeu Sim:

Que tipo de aplicações usa com as tecnologias referidas?
Navegação no sistema___ Escrita de mensagens___ Jogos___

Outras:_____

Usa: Várias horas por dia___ várias horas por semana___ várias horas por mês___
Várias horas por ano___ outro tipo de utilização:_____

2. Opinião geral sobre o *DETI Interact*

Após a utilização do sistema e tendo em conta a sua avaliação final, preencha o círculo que melhor reflecte a sua opinião em relação à utilização da ferramenta. Caso considere que estas quantificações não são aplicáveis, escolha NA.

2.1. Opinião sobre a utilização do *DETI Interact* *(preencha o círculo da opção que melhor corresponde à sua posição)*

2.1.1 Interacção através do **Smartphone:**

É fácil orientar-me no *DETI Interact*.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A velocidade de navegação é adequada.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA

A navegação é agradável.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A navegação tem algumas características irritantes.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A utilização do *DETI Interact* com esta forma de interacção é intuitiva.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A utilização do *DETI Interact* com esta forma de interacção requer treino.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A utilização do *DETI Interact* com esta forma de interacção é intuitiva.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA

2.1.2 Interacção através do Computador **Touch:**

É fácil orientar-me no *DETI Interact*.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A velocidade de navegação é adequada.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A navegação é agradável.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A navegação tem algumas características irritantes.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A utilização do *DETI Interact* com esta forma de interacção é intuitiva.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A utilização do *DETI Interact* com esta forma de interacção requer treino.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A utilização do *DETI Interact* com esta forma de interacção é fácil.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA

3. Por favor indique o grau de satisfação na utilização de cada uma das formas de interacção numa escala de 1 (muito baixo) a 5 (muito alto):

Smartphone:	1 nada satisteito	○	○	○	○	○	5 muito satisfeito

Computador Touch:	nada satisteito	○	○	○	○	○	muito satisfeito

4. Por favor assinale com qual das formas de interacção preferiu navegar no *DETI Interact*:

Smartphone: ☐ ☐ ***Touch***

5. Se pretender pode deixar aqui comentários sobre as formas de interacção usadas no *DETI Interact*:

_____
_____
_____
_____
_____

6. Que outros serviços gostaria de ver integrados no DETI Interact.

_____
_____
_____
_____
_____

FIM

Muito obrigada pela sua colaboração!

palavras-chave

Interacção Humano-Computador, Ecrãs públicos, Ecrãs de grandes dimensões, Usabilidade, Computação móvel

resumo

O uso de ecrãs de grandes dimensões para a exibição de conteúdo informativo contém limitações a nível de interacção que devem ser superados. Os dispositivos móveis actuais, equipados com acelerómetros, bússolas digitais, e ecrãs sensíveis ao toque, constituem uma alternativa para permitir esta interacção sem a necessidade de hardware adicional, uma vez que os utilizadores transportam o periférico consigo.

Esta dissertação apresenta os passos necessários ao desenvolvimento do DETI Interact: um sistema informático para exibição de conteúdo informativo para o Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro, que permite a utilizadores usarem um dispositivo móvel Android para interagir com os conteúdos apresentados. Após o desenvolvimento foram realizados testes com utilizadores e o resultado final foi colocado em funcionamento nos ecrãs de grandes dimensões presentes na entrada do departamento.

keywords

Human-Computer Interaction, Public Screens, Large Screen Displays, Mobile Computing, Usability

abstract

The use of large screen displays for the presentation of informative content has interactivity limitations that should be overcome. Current mobile devices come equipped with accelerometers, digital compasses, and touchscreens that constitute an alternative to allow interaction with these displays without the need for additional hardware, since the users will be carrying the peripheral with them.

This dissertation presents the necessary steps in the development of DETI Interact: an information system for the exhibition of informative content from the Department of Electronics, Telecommunications, and Informatics of the University of Aveiro, which enables users to use an Android mobile device to interact with the displayed contents. After its development, it was submitted to user testing and it is now operating in the large screens present in the department.

Table of Contents

Index of Figures

Acronym List

API Application Programming Interface

ASP Active Server Pages

CLR Common Language Runtime

CPU Central Processing Unit

DETI Departamento de Electrónica, Telecomunicações e Informática

DSD Distribuição de Serviço Docente

FCL Framework Class Library

GDI Graphics Device Interface

GPS Global Positioning System

GPU Graphical Processing Unit

HTTP Hypertext Transfer Protocol

IDE Integrated Development Environment

IIS Internet Information Services

LCD Liquid Cristal Display

LINQ Language Integrated Query

MFC Microsoft Foundation Class

MVVM Model-View-View Model

NFC Near-Field Communication

NUI Natural User Interface

OEM Original Equipment Manufacturer

OS Operating System

SDK Software Development Kit

UI User Interface

URI Uniform Resource Identifier

URL Uniform Resource Locator

UX User Experience

W3C World Wide Web Consortium

WCF Windows Communication Foundation

WPF Windows Presentation Foundation

WSDL Web Services Description Language

1 Introduction

1.1 Motivation

As technology evolves, computer and television screens have become increasingly thinner and cheaper. In addition, some of these screens are even equipped with touch-sensitive panels enabling interaction via touch input. This has allowed these screens to be used in public locations such as airports, train stations, malls, museums, lobbies of buildings, etc., presenting relevant information to the people passing by. The use of these screens, however, presents a challenge in how information is displayed and how people can interact with it. When the screens are touch-sensitive it is relatively easy to make the system interactive, however, when they are not, the system usually displays a predetermined set of information, usually presented in a looping or alternating sequence. The objective behind DETI Interact is to study how to make these screens interactive, allowing the user to control the flow of information and choose what he wishes to see.

Interaction with these screens can be achieved in a variety of ways, as can be seen in several projects presented in Chapter 2. These projects research the use alternative input peripherals, such as a smartphone, to more advanced scenarios such as video tracking or motion capture. DETI Interact is a follow-up project to DETI Guide (Palha 2010), a previous study that investigated how a smartphone could be used to interact with a public screen, for this reason, the chosen approach was to use a smartphone. The result is that there will be no hardware changes required on the system (for instance, no tracking cameras, touch panels, etc.), since users will be carrying the interaction peripheral themselves. However, the choice of how interaction is done is only one of the steps.

Given the nature of the device, several aspects concerning the interaction have to be addressed with care: the way the content is displayed on the screen, the way the user interface is designed, and the way the device interacts with the content, are all important factors, which will be addressed in this dissertation.

1.2 Objectives

DetiGuide(Palha 2010) was a system developed in the scope of a previous dissertation, which studied the possibility of using of a smartphone to interact with the screens in the lobby of the Department of Electronics, Telecommunications, and Informatics of the University of Aveiro. The present dissertation intends to go further. Following the results from the previous study, this project aims to develop of a platform for displaying interactive information on those screens, and integrate the possibility of interacting with this information via a smartphone.

This project has two main objectives:

- System architecture: the existing DSD (*Distribuição de Serviço Docente*) Platform which supplies information to the system that is currently operating in the lobby of the department should be studied in order to use the supplied information to develop interactive contents.

- Interaction with the displays: the developed system should allow a user in the lobby to interact with the screen to access relevant information, such class times, information regarding the teachers, etc. In addition to planning the system for communication with mobile devices, the system should also be prepared to communicate with other devices, such as a WiiMote or cameras via body tracking. It should also be prepared to run on computers with touch-sensitive screens.

The resulting developed system will be deployed to the computers in the lobby of the department where users will be able to interact with the system using their phones.

1.3 Dissertation Structure

This dissertation consists of five chapters, excluding this introductory section.

In chapter 2, “State of the Art”, the bibliographic research done for the development of this system is presented. It encompasses several projects that propose different methods of interaction between users and large screens.

Chapter 3, “System Design”, illustrates the different steps in the preparation of the development of the system. It describes the used tools, studies features that were chosen to integrate the system, as well as some of the required APIs, and presents the architecture of the system, leading to the next chapter.

In Chapter 4, “Development”, the development phase of the project is described. Some implementation details as well as problems are explained in order to illustrate how the core of the system works, and how it can be modified or extended.

Chapter 5, “User Evaluation”, explains how the developed system was tested. The results of the experiments are presented, and are discussed as to how they are used to improve the system.

Finally, in Chapter 6, “Discussion”, the work is reviewed, along with the results gathered from the testing sessions. Some possibilities for future work on the developed system are discussed.

2 State of the Art

2.1 Introduction

With advances in technology making computer screens increasingly thinner and cheaper, these became more common in public places such as shops, cafes, museums, atriums of buildings, and train and subway stations. However, these screens are often used to display information, rather than allowing users to interact with it. These public displays fostered several research projects to investigate interaction between users and the displayed information using mobile devices, augmented reality, and motion capture.

The projects studied for the development of DETI Interact propose diverse ways for interaction with information presented on computer screens. Apart from the many differences between these projects, some recurring elements allowed these studies to be grouped into three distinct categories. The first group of projects focuses the use of smartphones for interaction, concentrating on the technologies that can be used for communication between devices such as Wi-Fi, Bluetooth, and SMS. They present different methods of communicating data between mobile devices and large screens in order to interact with some part of the system. The second group of projects uses different approaches of video tracking and augmented reality to allow interaction between mobile devices and the target screens. Finally, the third group uses motion capture for the interaction, eventually dismissing the use of mobile devices altogether.

2.2 Communication between devices

Since DETI Interact will involve communication between a mobile device and a computer, the several means of achieving this communication must be studied, allowing for a more educated decision.

Gmote (Stogaitis and Sun 2008) is an example of an Android application that enables interaction between a mobile device and a computer, using a Wi-Fi connection. For the devices to communicate an application that acts as a server is deployed to the computer and a client application is executed on the mobile device. This system supports most of the media playback related features found in a standard remote control such as play, pause, rewind, volume control, etc. as well as new features like file browsing and media streaming. It can also use the mobile device's touchscreen as a touchpad, as well as Android's software keyboard as a keyboard, enabling the user to control the mouse pointer and keyboard remotely. Similar applications such as RemoteDroid, BoxeeRemote, TivoRemote, and SqueezeControl, have also been developed that allow interaction between additional mobile and desktop operating systems (Wired 2009).

While Gmote can be used to interact with a computer, similar applications have been developed that allow interaction with a television or a set-top box. Samsung has developed the Internet@TV platform (Samsung 2010), a service that allows users to access online content such as photos, video, music, social networks, news, games, etc., on an internet-enabled Samsung TV or Blu-ray player. Interaction with the system can be done with a regular remote control, however, Samsung has developed an application for iOS and Android-based devices, which lets users interact with the television or set-top box using their mobile phone. Google has also developed a similar system, Google TV (Google 2010), which also supports the use of a phone to control the device. In addition to typical functions of a remote control, Google's phone application also enables streaming video and audio content to the television.

Other systems, such as Touch & Interact (Hardy and Rukzio 2008; Hardy and Rukzio 2008), allow users to interact with large displays using mobile devices with NFC (Near Field Communication) technology. NFC consists of a set of short-range wireless technologies, such as RFID, typically relying on radio frequency for communication. The display used in this system is attached to a mesh of NFC tags, which enables the phone to detect the tags when close to the display. Given the nature of NFC tags, the system supports gestures such as hovering the phone over a certain tag, or touching an area close to the tag. The phone is used to identify the tags on the display, and communica-

tion with the system is done using Bluetooth. Figure 2.1 shows a user selecting cells on a map application on a display using Touch & Interact.

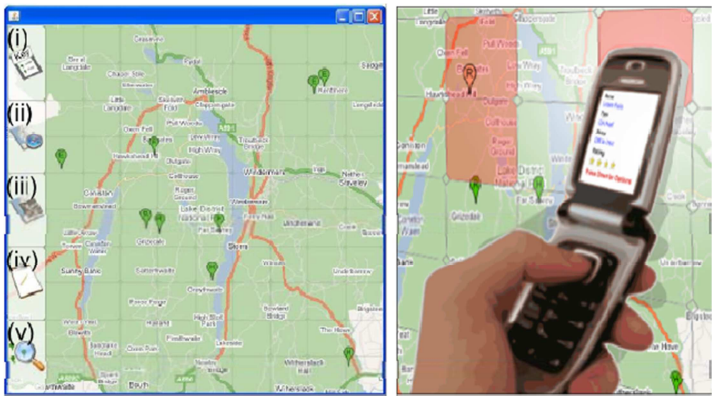

Figure 2.1 - A tourist guide application using Touch & Interact (Hardy and Rukzio 2008).

The principle behind Touch & Interact also produced two additional projects: Touch & Connect and Touch & Select (Seewoonauth, Rukzio et al. 2009). Touch & Connect allows users to transfer files between a laptop and an NFC phone simply by touching the NFC tag placed on the laptop. The system then configures the connection and transfers the date, without any additional user input required. For Touch & Select, a mesh of NFC tags similar to the one used in Touch & Interact allows the user to select items on the laptop by touching its display with the mobile device. Communication with the system in either case is done via Bluetooth.

MAGICBoard (Tang, Finke et al. 2008) is a public digital forum designed to encourage bystanders to interact with the system. Bystanders are individuals that are close to the large display but never actively participate in using the system. The system allows users to view or discuss on popular topics, and participate in some voting sessions. Interaction with the system was achieved by using either SMS messages, or a nearby kiosk terminal. An example of a deployed MAGICBoard application can be seen in Figure 2.2. Polar Defence (Finke, Tang et al. 2008) is a tower defence game that was deployed in a similar setting, using the same system behind MAGICBoard. Users would play the game by sending SMS messages to the system with the appropriate instructions. These projects attempted to reduce barriers to interaction, such as embarrassment, trust and security, and encourage spectators to become actors in the system (Kaviani, Finke et al. 2009).

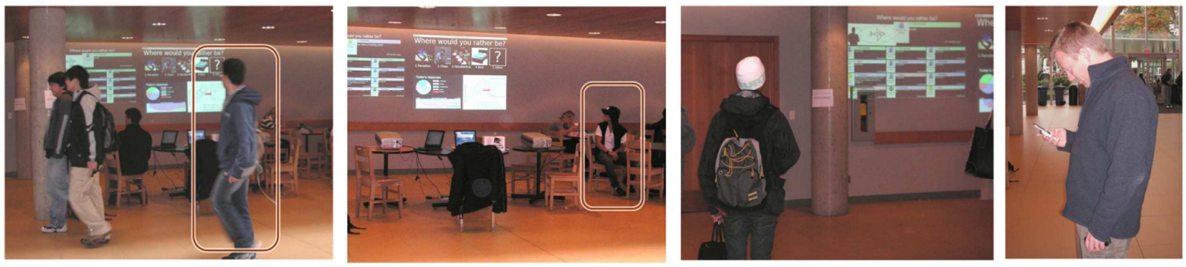

Figure 2.2 – A deployed MAGICBoard (Tang, Finke et al. 2008) system illustrating, from left to right, a *passer-by*, a *stander-by*, and *engaged bystander*, and a *contributor*.

As mentioned in Chapter 1, DetiGuide (Palha 2010), a project that preceded DETI Interact, studied interaction with large displays using a mobile device running Android OS. The system was deployed to computers located at the lobby of the Department of Electronics, Telecommunications, and Informatics of the University of Aveiro, and consisted of a web application similar to the department's website, and a mobile application, used to interact with the web application. The web application displayed information regarding how to navigate in the system through a series of widgets arranged along the edges of the web page. This information would assist the user in the manipulation of the mobile device in order to navigate on the web site. The mobile application supported several interaction techniques, such as using the accelerometer, gestures drawn on the touchscreen, and methods to control the mouse pointer. Communication between the mobile device and the web application was done over a Wi-Fi connection.

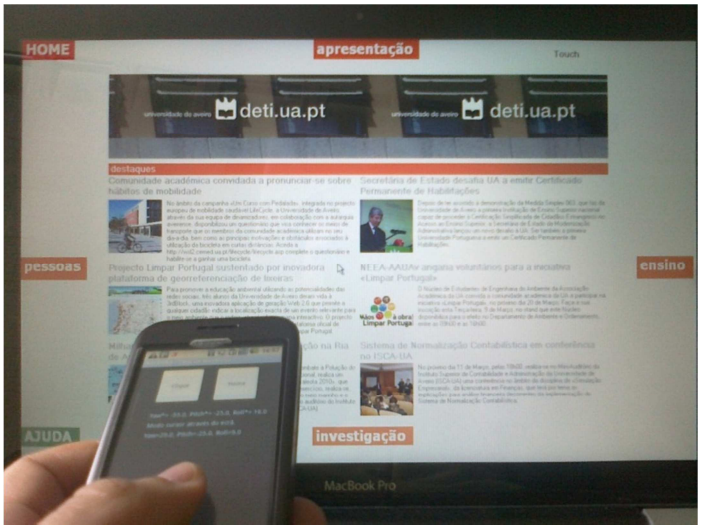

Figure 2.3 – User interacting with DetiGuide using the accelerometer on a mobile device.

In conclusion, the most used communication methods include Bluetooth, Wi-Fi, or SMS. For DETI Interact, Bluetooth and Wi-Fi present the most versatile alternative since there is no interference from the operator in the communication process.

2.3 Video Tracking

Interaction with large displays can also be achieved by tracking objects on a video feed, such as the movement of light sources. LasIRPoint (Cheng and Pulo 2003) uses an infrared tracking device and an infrared laser pointer in a presentation environment. The tracking device is coupled with the projector, facing the screen, so that it captures the gestures performed by the user with the laser pointer. Users can perform gestures such as selecting an object by drawing a circle around it with the laser pointer, and the system processes these gestures and proceeds accordingly.

C-Blink (Miyaoku, Higashino et al. 2004) uses a camera to track the movement of a mobile phone with a coloured screen (Figure 2.5). The phone runs an application that rapidly changes the hue of the LCD screen; this hue sequence is unique and easy to detect, and it can be used to identify multiple mobile phones, which allows the system to support multiple users. The camera is positioned above the display, and the user waves the phone in front of it (Figure 2.4). It uses the signal coming from the phone to track its position and uses this to control the position of the cursor on the display.

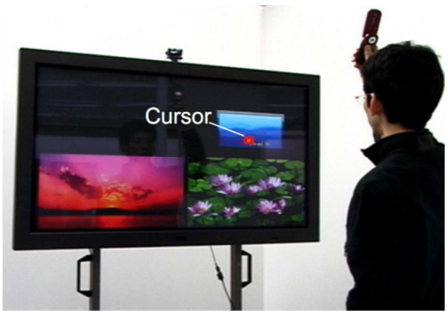

Figure 2.4 – Using C-Blink to control a display (Miyaoku, Higashino et al. 2004).

Figure 2.5 – C-Blink with an overlay of the video captured by the sensor (Miyaoku, Higashino et al. 2004).

In addition to identifying the user, a C-Blink signal can also encode extra information. To illustrate this, the system also implements several basic functions such as Click, Grab, and Pitch. Each of these functions is sent to the system through a C-Blink signal. For instance, if a user wants to download an image from the system (using the Grab function) he aims the device at the image and presses a button; this generates a C-Blink signal that encodes the terminal ID of the cell phone and along with a flag for the grabbing function. The system then responds by sending the image to the phone via the wireless network.

Touch Projector (Boring, Baur et al. 2009) is a system that uses an augmented reality solution allowing users to remotely manipulate content presented on remote displays on a mobile device via a live video image captured by the device's camera. The user aims the mobile device at a display and interacts with the image on the screen via touching and dragging the objects. Input to the touchscreen is "projected" to the selected display as if it had occurred there. The user can select an object and move it with a finger (Figure 2.6), which results in a relatively precise movement, or he/she can select and hold an object and move the device instead, which performs a more coarse movement. Objects can also be moved between displays by touching the object in the first display and moving the device towards the second, dragging the object off-screen and into the intended display.

Figure 2.6 – Selecting an object using Touch Projector (Boring, Baur et al. 2009).

In order to achieve a better user experience, features such as zooming and freezing the video image are also available. Zooming allows the user to interact with distant displays; this can be done automatically when the user aims at a display where the image does not fill the whole area of the video. Freezing the image allows for higher precision since the user will be dealing with a still image; it also eliminates the need to keep the device still or pointed at the screen, which avoids fatigue.

Madhavapeddy *et al.* (Madhavapeddy, Scott et al. 2004) also present a camera-based interaction using mobile phones, involving manipulating tagged interactive UI elements such as sliders and dials (Figure 2.7). Interacting with the system relies on using *SpotCodes* (Figure 2.8), a visual tag similar to a QR-Code, which is detectable by the camera-phones. These tags can be active (e.g. generated dynamically by the system) or passive (e.g. printed on paper). By placing the phone close

to the tag, ensuring the camera on the phone can identify it, the user can position a graphical slider, or orient a graphical dial by manipulation the camera-phone.

The mobile phone is responsible for tracking, and identifying the tags, while the system is responsible for displaying and refreshing the tracked images. Communication between the devices is achieved using Bluetooth.

Figure 2.7 – Madhavapeddy's *et al.* (Madhavapeddy, Scott et al. 2004): User interacting with a world map application.

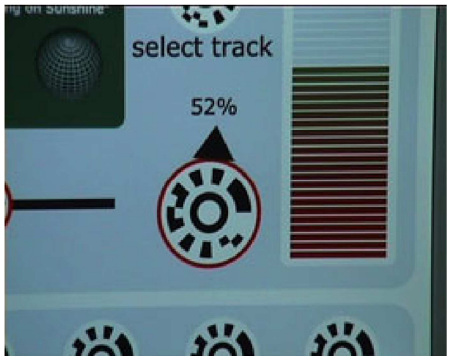

Figure 2.8 - Example of a dial tagged with a *SpotCode* using Madhavapeddy's *et al.* (Madhavapeddy, Scott et al. 2004) system.

2.4 Motion Capture

Other ways used to interact with information displayed on large screens include motion capture techniques. Motion Capture systems typically require specific cameras, such as Infrared or Stereoscopic cameras, as well as markers on the tracked objects to assist in the motion capture process.

Vogel *et al.* (Vogel and Balakrishnan 2004) developed a prototype system for use in interactive public ambient displays which allows for multiple users. Interaction with the system is performed using simple hand gestures (Figure 2.9) as well as touch screen input. A motion capture system is responsible for the detection of users and gestures, and requires the use of small passive markers on the body parts that require tracking. While this is inconvenient, advances in computer vision techniques should obviate the need for markers.

Figure 2.9 – Users interacting with the prototype system developed by Vogel *et al.* (Vogel and Balakrishnan 2004).

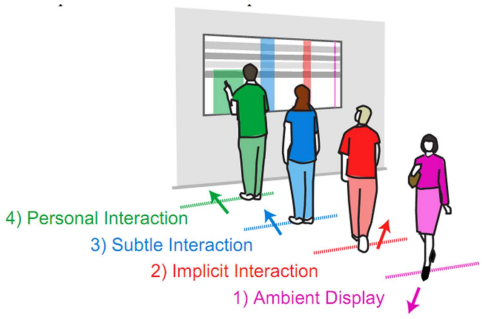

Figure 2.10 – Illustrating the four interaction zones covered by the system(Vogel and Balakrishnan 2004).

This prototype explores the concept of transitioning from implicit to explicit interaction with both public and private information. The system covers four interaction zones: Personal Interaction, Subtle Interaction, Implicit Interaction, and Ambient Display, as shown in Figure 2.10. When a user is far from the screen, the system will perform as an ambient information display, showing a range of categorized information. If the system detects that the user is closer and facing the screen, the system enters the Implicit Interaction phase, adapting the UI by displaying brief information concerning the user, as well as subtle notifications regarding personal or public information items that may require attention. As the user approaches the screen, the system enters the Subtle Interaction phase; the system adds more detail to the information and notifications onscreen. When the user selects an information item the system enters the Personal Interaction phase, the user is now in front of the screen, which allows for direct manipulation of the information.

Underkoffler (Underkoffler 2010) developed a system in which users would wear a set of gloves with reflective material placed on the tips of the fingers. The system would then be able to track the movement of each finger, which would allow the user to interact with large displays using hand gestures. This system was developed to be used on the sci-fi movie Minority Report, and has been updated to allow interaction with three-dimensional environments.

Orange, the UK based mobile network operator, unveiled a gesture based interaction screen, developed by The Alternative (TheAlternative): the Orange Interactive Window (GIZMODO) (Figure 2.11). The system uses cutting-edge motion capture technology that tracks gestures executed by the users to control the content on the screen without the need to place passive markers on the user (Figure 2.12). Other competitors in the mobile market such as Vodafone, Samsung, and Sony, have also developed their own versions of this interactive window.

Figure 2.11 – The Orange Interactive Window (GIZMODO): Main display.

Figure 2.12 – User interacting with the Orange Interactive Window (GIZMODO).

Kinect (Microsoft 2010), developed by Microsoft, also offers state-of-the-art motion capture technology. It is intended for home use on the Xbox 360 video game platform and eventually computers equipped with Microsoft Windows (Anthony 2010).

Figure 2.13 – The Kinect sensor device (Microsoft 2010).

The device, shown in Figure 2.13, includes an RGB camera, a depth sensor and a microphone array (Totilo 2010). The RGB camera is used as a regular video camera, and can be used for facial recognition. The depth sensor consists of an infrared laser emitter and a CMOS Active-Pixel Sensor. The emitter projects a mesh of infrared points to the surrounding environment, which the sensor captures; the distance between these points is used to tell how far a certain object is. By mapping the depth image with the image from the RGB camera, the system can create a 3D video image of the surroundings. This allows the system to provide 3D full-body motion capture under any ambient light conditions. The microphone array can suppress ambient sounds, and locate sound sources, which enables innovative voice recognition capabilities.

Figure 2.14 – Using Kinect with an application in Windows 7.

Figure 2.15 – Two users interacting with Kinect simultaneously.

The hardware behind Kinect along with its software technology enables the system to track up to six people simultaneously, including two active users (Figure 2.15). The system can recognize 31 body parts per user, which allows for the detection of gestures of a certain complexity. Since launching the device, Microsoft has revealed plans to integrate Kinect with Windows (Figure 2.14) (Anthony 2010), and has released a beta version of its software development kit to the public (Microsoft 2011) enabling developers to create Windows applications that target the Kinect device. This differentiates Kinect from its competitors developed by Sony and Nintendo.

3 System Design

In this chapter, the choices dealing with the definition and the design of DETI Interact will be detailed. The content to be presented by the system will be specified, as well some requirements to present it. This will also lead to the evaluation of development platforms. Finally, the system architecture will be explained.

3.1 Introduction

The Department of Electronics, Telecommunications, and Informatics (DETI) of the University of Aveiro has, in its lobby, three large screens. These screens display a website that contains information related to the department, specifically the most recent news, and a list of all the lecturers in the department. The information contained on these webpages is stored in a server running the DSD Platform (*Distribuição de Serviço Docente*).

DETI Interact intends to replace the existing system by displaying additional information and allowing users to interact with it, enabling navigation and selection operations in the system. Interaction with the system will be done via mobile devices powered by the Android operating system. Communication between devices must be achieved seamlessly by the system on behalf of the user.

In order to conduct a better evaluation of the system, interaction with the system with a mobile device will be compared with interaction with a touchscreen, as well as a mouse. To ensure some fairness in the comparison, the system will include different types of content and methods to interact with it that better leverage the advantages and disadvantages of each device. This content includes information from the university's servers (regarding lecturers and timetables), as well as ma-

nipulation of a map from Google Earth, and three-dimensional content. Each of these sources of data has its own method of integration with the system, such as a dedicated API (Application Programming Interface) or a series of Web Services. This requires that each source be studied as to how it can be included in the system, which will in turn affect the development environment, since some APIs may be exclusive to a certain platform or programming language.

3.2 System requirements

The selected content to be presented on the initial version of DETI Interact includes content from the department's DSD Platform, as well as a map from Google Earth, and three-dimensional content. The DSD platform holds information relevant to the department, thus it will be the source for this data. From the DSD server information regarding the department's teachers and timetables will be gathered. Google Earth offers an intuitive way to use the touchscreen on the mobile device, via gestures like scrolling or pinch to zoom, making it a fitting addition to the system. The use of the digital compass and the accelerometer on the mobile device brings an advantage to the presentation and manipulation of three-dimensional content making it another suitable addition.

The DSD Platform, Google Earth, and existing 3D APIs must be studied to determine how these can be implemented in DETI Interact.

3.2.1 DSD Platform

The DSD Platform (Campos 2008) is a system that allows lecturers and students to perform operations related to the academic year. It is actively used every year by students on situations such as checking timetables, and enrolling in subjects. Students can also make choices regarding dissertation proposals, which are added to the system by the lecturers. It is, essentially, a management system for the department.

The system consists of a server computer running a Microsoft IIS (Internet Information Services) web server, which is hosting an ASP.NET web application that allows remote access to the system. The main webpage for this application can be seen in Figure 3.1. It also contains a SQL database managed by Microsoft SQL Server. The database holds all the data relevant to the department such as lists of students and lecturers, all the lectured courses along with the corresponding year and semester, class times and classroom occupation, etc.

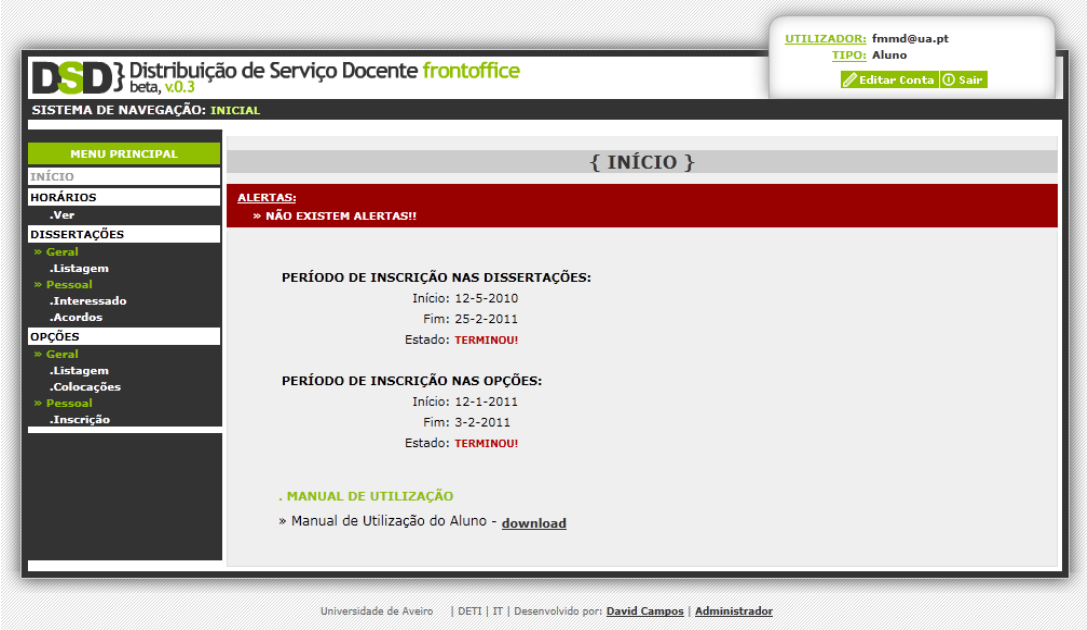

Figure 3.1 – The DSD Platform's web page.

Some of the information in the database can be accessed via a series of Web Services published by the ASP.NET Web Application. A Web Service is, as defined by the W3C, "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards" (W3C 2004).

Web services allow programmers to easily access the DSD platform's API, meaning that DETI Interact can collect and handle data from the server. The existing Web Services are quite limited in what concerns to the type and amount of information they return, but they provide enough data for some basic operations. The data returned can be used, for example, to recreate the existing list of teachers, as well as evaluate classroom occupation, generate timetables, etc. Furthermore, should these Web Services be insufficient, new ones can later be developed if the need arises. The list of available Web Services is as follows:

- getAreas();
- getAulas();
- getCursos();
- getDepartamentos();
- getDisciplinas();
- getDisciplinasAno();
- getDocentes();
- getDocentesAno();

- getSalas();
- getSemestres();
- getSubAreas();
- getTiposSala();
- getTurmas();
- getTurmasHora();
- getWishList();

3.2.2 Google Earth

Google Earth (Google 2005) is a virtual globe application, with mapping and geographical features. It maps the earth's surface by overlaying images obtained from satellite imagery and aerial photography. The product was initially released as a standalone, downloadable application. However, it has since become accessible via a web browser plugin.

Introducing Google Earth to DETI Interact provides a way of comparing different methods of interaction. Given its graphical user interface, it constitutes a suitable choice to compare interaction with mouse and keyboard, with interaction using a touch screen on a mobile device. Google provides APIs for developers to use its mapping services, making integration of the Google Earth plugin possible. This API, however, is only accessible through Javascript, which is intended for use on web applications. Including Google Earth on the system involves finding a way to parse HTML and Javascript and displaying the results within the application. This can be done by either accessing Web Browser controls or using libraries on the OS for that purpose.

3.2.3 3D Content

Manipulation of three-dimensional content represents another appropriate choice for the comparison of different interaction devices. The use of compass or accelerometer on mobile devices can be compared with the use of the mouse and keyboard, or a touchscreen, on a computer.

There are numerous ways to display three-dimensional content on a computer, including OpenGL, DirectX, VTK, and XNA. These graphical APIs are used to produce standalone applications, but can also be used to display three-dimensional content within applications, albeit with some minor adjustments. The use of any graphics API will depend on the development platform: OpenGL and VTK are cross-platform, DirectX is restricted to C++ development on Windows, although there are managed wrappers for .NET, and XNA is a Windows exclusive as well.

In order to use any of these APIs to render a scene within a desktop application it is necessary to either use an available wrapper, which allows the developer to create a user interface around the three-dimensional content, or redirect the target render buffer to a different container (e.g. an image), and display this container in the application.

3.2.4 Mobile Devices

In order to target mobile devices it is necessary to decide whether to develop a web application, enabling all the mobile platforms to interact with the system, or to develop a native application, which targets a specific platform.

The ideal solution would be to develop a Web Application for mobile devices instead of a native application since it would be able to target any mobile device with a web browser, regardless of operating system type or version. It also provides better control over the application: there is only one application to develop and update, and every user will always be using the latest, most up-to-date version. However, the problem with web application would be accessing the device's hardware (Fling 2009). For DETI Interact, apart from the touchscreen, it is also useful to have access to hardware such as the accelerometer and Bluetooth adapter. In addition, access to the camera, GPS, storage, etc., might be necessary in future versions of the application.

Another solution for distributing for multiple platforms would be using PhoneGap (PhoneGap 2008). This mobile framework currently allows development for six different mobile platforms: Apple iOS, Google Android, Blackberry OS, Palm webOS, Windows Mobile, and Symbian. It provides access to hardware such as the accelerometer, camera, compass, GPS, and storage, but still offers no access to the Bluetooth adapter. Navigation in the applications built using PhoneGap also are not consistent with the operating system, for example, pressing the 'back' button on an Android device exits the application, instead of navigating back to a previous screen.

Since the mobile application for DETI Interact requires access to hardware unavailable when developing for both the mobile web and PhoneGap, the remaining solution is to develop a native application. Given that the Global market share for the Android platform has revealed a rapid growth and a high adoption ration, quickly becoming the leading mobile platform (DailyTech 2011) this was the targeted platform for the initial version of DETI Interact. In addition, there was already Android-powered hardware from previous studies, available to be used for this system.

Developing for Android is done via a provided SDK that allows access to all the required hardware, which is unavailable or limited in other platforms such as iOS or Windows Phone 7. Other mobile platforms can easily be addressed at a later stage.

3.3 Development Environment

With the specification of which content is to be presented, steps can be taken into the development of DETI Interact. Given that the DSD Platform is developed using the .NET framework, and Android devices require the use of the Java platform, the use of either of these environments to develop DETI Interact would be beneficial given the vast amount of existing libraries for each platform. Content such as Google Earth and 3D scenes can be easily included in an application developed in any language.

Since the target for the mobile application is an Android device, the Java platform could also be used to develop the application that will be running on the large screens. One big advantage of using Java would be its platform-independence, since the Java Virtual Machine is deployable to most known platforms. Being such a widespread platform, a vast amount of libraries, wrappers, and documentation is also available. Additionally, the Java Class Library includes a Bluetooth API, which would be useful for communication between computers and Android devices. On the other hand, while its performance has improved substantially over time, Java still displays poor performance in some scenarios, as well as a high consumption of system resources (Mihailescu 2010). Furthermore, regarding security, Java has become the most common target for malware, surpassing Adobe Flash and Adobe Reader (Cull 2010). This could represent a security issue since the system is to be deployed at a public location.

An alternative for the Java platform is the .NET Framework (Microsoft 2009). While it cannot match Java in platform independence, .NET still covers a wide range of platforms thanks to the Mono Project that implemented its own version of the Runtime; Mono's implementation of the framework, however, lags behind Microsoft's in some areas. The framework developed over time and now includes features unavailable on the Java platform such as Language Integrated Query, for data processing, and Windows Presentation Foundation, an innovative tool used for user experience and user interface design. In terms of performance, while .NET (and managed applications in general) will probably never be as fast and memory efficient as their native counterparts, applications tend to be faster, due to its Just-In-Time Compiler, and less resource intensive, due to its Garbage Col-

lector (Mihailescu 2010). In Addition, .NET assemblies also take advantage of Windows' Data Execution Prevention and Address Space Layout Randomization, which improves the overall security of the system (Richter 2010).

While communication via web services is independent of platform or language, consuming .NET Web Services from a different platform, such as Java, is not as streamlined as doing so from within the .NET platform. Google Earth and 3D content can be implemented in either platform. Furthermore, communication between the Android device and the desktop application via Bluetooth is done through the exchange of packets of data, such as OBEX messages, which is independent of platform or language. Taking in to account the pros and cons of each alternative, the chosen development environment for the computer application was the .NET Framework.

3.4 Development Tools

3.4.1 Software

In order to develop DETI Interact, two applications must be developed: a desktop application, in .NET, and an Android application in Java. Various tools were used to assist and speed up the development of these applications.

3.4.1.1 Microsoft Visual Studio 2010

Since the system would be built on the .NET platform, using Visual Studio was an understandable choice. Microsoft Visual Studio 2010 is the most recent version of this IDE (Integrated Development Environment). It ensures quality code throughout the entire application lifecycle, from design to deployment, and assists in the development of applications for Windows, Windows Phone, SharePoint, Web, Cloud, etc.

Visual Studio was used to develop the portion of the system that ran on the computers at the lobby of the department. The application along with some addition libraries was developed in C#, and the user interface was built with XAML, used by Windows Presentation Foundation and Silverlight.

3.4.1.2 Microsoft Expression Blend 4

Expression Blend is a user interface design tool belonging to Microsoft's Expression Studio suite. It is an interactive front-end for designing XAML-based interfaces for Windows Presentation Foundation, Silverlight, and Windows Phone applications (Microsoft 2011). Blend allows developers to focus on the user interface design of the applications, without affecting the business login layer. It promotes the usage of the MVVM architectural pattern.

3.4.1.3 Eclipse

Eclipse is a multi-language, multi-platform software development environment. It consists of an Integrated Development Environment and an extensible plug-in system. This plug-in system allowed for the development of the Android Development Tools (ADT) plug-in which includes a device emulator, and tools for debugging, and memory and performance profiling. It is the IDE recommended by Google, and was used for the development of the Android application.

3.4.2 Hardware

DETI interact is expected to run on a computer connected to a large display as well as a computer equipped with a touchscreen. These are both Asus machines: an Eee Box, connected to the large display, and an Eee Top. The mobile application that will interact with the system will be tested on a provided Motorola Milestone.

3.4.2.1 Asus Eee Box

The Asus Eee Box is a small form-factor desktop computer. It has a low-end processor and integrated graphics card, which allows it to consume very little power. This, however, can have a negative impact on the performance of the application, since the system will deal with very large sets of data, as well as three-dimensional content.

3.4.2.2 Asus Eee Top

The Asus Eee Top is a computer similar to the Eee Box line, but its hardware is built into a touch-sensitive screen. In addition to the low-end hardware, the screen only supports single-point interaction, which will limit the set of gestures that can be used as well as affect the overall user experience.

3.4.2.3 Motorola Milestone

The Motorola Milestone is a quad-band version of the Motorola Droid, an Android-powered smartphone. The hardware is similar between the two, with only minor differences in internal memory and installed applications. It was the device used to test the developed mobile application, and was also used for the tests performed with users. During the time the DETI Interact was deployed to the computer in the lobby, users were able to install the Android application in their personal devices.

3.5 DETI Interact Design

With all the system requirements taken care of, a development environment chosen, and all the hardware detailed, the system architecture as well as some design and development considerations will be described.

3.5.1 Architecture

As discussed previously, DETI Interact will consist of two applications. The first application will run on the desktop computers connected to the large displays. This application will contain the content gathered from the DSD server, the map from Google Earth, and a viewer for 3D content. To allow interaction from Bluetooth devices, a Bluetooth socket will be waiting to acquire a device. The desktop application will interpret data packets sent by the mobile device in order to perform operations such as changing the displayed content. While no device is connected to the system, the desktop application will be able to enter a demonstration mode, cycling through all the available content.

The second application will run on an Android mobile device. It will observe gestures drawn on the touchscreen, as well as the values from the digital compass or accelerometer, and send this data as a structured packet via Bluetooth to the desktop application. A diagram of the system architecture can be seen in Figure 3.2.

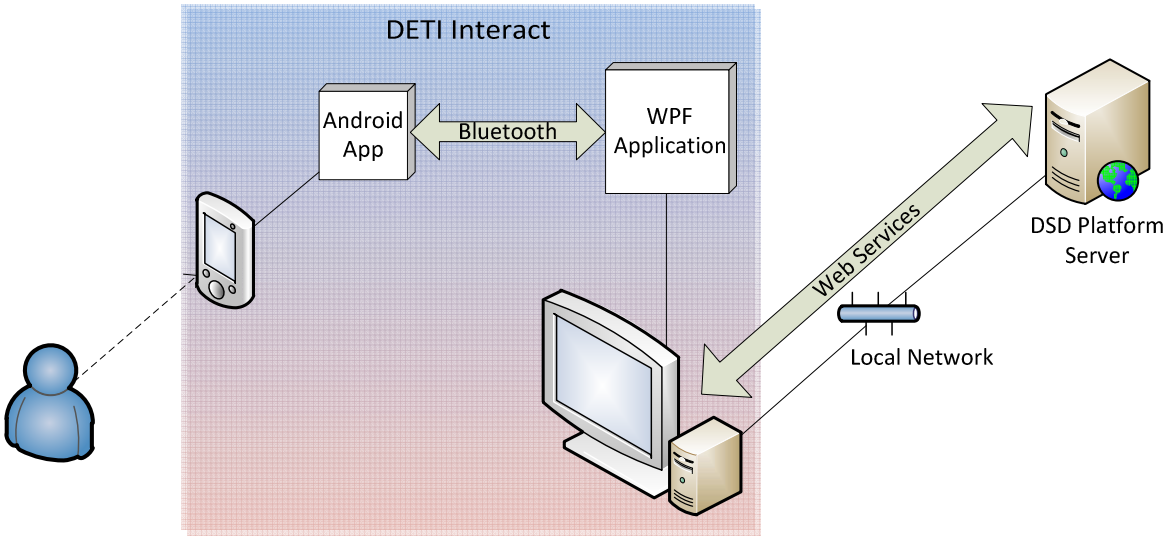

Figure 3.2 - DETI Interact architecture

3.5.2 Desktop Application – .NET Framework

As discussed previously, the .NET Framework was chosen as the development environment for DETI Interact. The mobile application, developed for Android devices, will be written in Java, and the desktop application, which will run on the computers on the department, will be written in C#.

The .NET framework is a programming environment developed by Microsoft for building, deploying, and running applications and services that use the .NET technologies. The framework consists of two main components: the Common Language Runtime (CLR) and the Framework Class Library (FCL). The CLR is a runtime that contains features such as memory management, assembly loading, security, exception handling, and thread synchronization, and is usable by various programming languages. Currently dozens of languages can target the runtime of which the most widely used is C#. This was the programming language used in the development of DETI Interact. It also includes features that are not available in other platforms but are useful for this system, such as LINQ and WPF.

LINQ – Language Integrated Query

Using .NET benefits DETI Interact when it comes to handling data from the DSD database due to the inclusion of LINQ. This feature adds native querying capabilities to .NET languages meaning

that the programmer can write code similar to SQL queries in order to parse data, instead of looping through all the results returned from the web services. This can speed up certain portions of the code, ensuring a more responsive application.

WPF – Windows Presentation Foundation

Another noteworthy addition to the .NET Framework is Windows Presentation Foundation (WPF). Microsoft describes WPF as being "a next-generation presentation system for building Windows client applications with visually stunning user experiences" (Microsoft 2010). Instead of depending on the older GDI subsystem like Windows Forms or MFC, WPF uses DirectX, which allows Windows to offload some graphics tasks to the GPU, enabling the CPU to work on tasks that are more significant. WPF also introduced a new architectural pattern: Model-View-ViewModel (MVVM). This pattern was created to make use of specific features in WPF and Silverlight, separating the View layer from the rest of the pattern. This separation of layers means that the business logic of the application is completely independent of the user interface.

3.5.3 Mobile Application – Android

As described on 3.2.4, the targeted mobile platform was the Android platform. Android is a software stack for mobile devices that includes an operating system, middleware and key applications (Google 2011). An SDK (Software Development Kit) is provided with the tools and APIs necessary to develop application on the Android platform.

Development for Android is done in Java, however, there is no Java Virtual Machine, instead the code targets the Dalvik virtual machine, which was specifically designed for Android and optimized for battery-powered devices with limited memory and CPU. The system integrates a web browser based on the WebKit engine, SQLite for data storage, and support for common audio, video and still image formats. It also features GSM telephony, Bluetooth, EDGE, 3G, WiFi, Camera, GPS, compass and accelerometer. All of these features are available in the platform's SDK.

The provided SDK allows developer access to the gestures drawn on the touchscreen, data from the compass and accelerometer, and the Bluetooth adapter. These are the necessary features for the system to work properly.

3.5.4 User Interface and User Experience

Designing a good user interface for the system and planning a good user experience are crucial to increase interest and to encourage users to use the system. The user interface is what allows the user to manipulate the application. It must provide effective ways for a user to operate and control the system. This should be done while providing a good user experience. User experience is a subjective concept since it varies between users. The system must be easy and designed in a way that the user does not feel the using the system is a chore. The user must feel that he/she knows how to operate the system at first contact, and can anticipate how the system will respond to his input.

There will be two very distinct screens in use: the large screen, and the mobile screen. This means that some care must be taken in how the screen will be used. Depending on how the screens display information, using both screens may cause user confusion. In order to avoid having the user diverting attention between the large screen and the mobile screen, the chosen approach was to have all the content presented on the large screen, and use the mobile device solely for interaction. This will allow the user to focus on the large screen, and forget about the mobile device, reducing the cognitive load on operating the mobile device correctly, in an attempt to resemble what happens with the computer mouse nowadays.

One other very important detail is how the information will be displayed on the screen. The available space must be used properly and it must provide a decent usage experience. Several aspects must be taken into account:

- The system will also be deployed to a touchscreen computer, which means that interaction with content on the screen must be finger-friendly;
- The large display is fixed on a wall, meaning that users will stand further away as they would from a regular computer, hence, content must be displayed in a larger format;
- Both screens have a widescreen format (with a 16:9 image ratio), with their width being almost twice as large as their height. Given the more limited height, screen space would be better used, for example, by stacking controls in a column rather than in a row (Figure 3.3), this, however, limits the amount of controls that can be stacked;

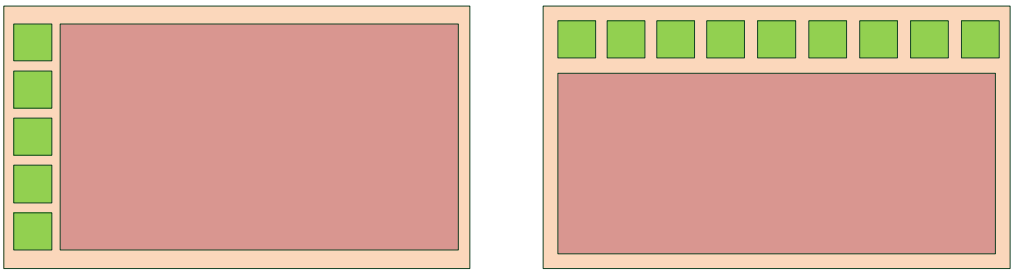

Figure 3.3 – Two distinct ways of using available space on a widescreen display.

- The screens will be manipulated by either mobile device or a touch screen, therefore the content, and interaction with it, must support a limited set of gestures.

Figure 3.4 - The original Xbox 360 dashboard (Thurrott 2010)

There are numerous ways of displaying content properly, each with their advantages. Media centres and videogame systems provide great user interfaces, directed for controllers with limited functionality namely directional and selection buttons. These systems provide a good starting point in the development of the user interface. The original dashboard included with the Xbox 360 video game system (Figure 3.4) provides a large amount of screen space for information, with different panes that can be used to switch context. This dashboard is derived from the UI used in Windows Media Centre, and newer versions of this dashboard have eventually blended with it. The concept behind these user interfaces has been in constant evolution for over a decade culminating in devices such as the Zune player and the Windows Phone (Thurrot 2010).

The original Xbox dashboard served as inspiration in the design of the user interface for the desktop application of DETI Interact. With information arranged in a similar manner a greater portion of screen can be used to display information, while the edges show which other panes area accessible.

3.6 Conclusion

In order to make DETI Interact a system appealing to the user, a series of features were chosen to integrate it. Communication with the DSD platform is required to provide content relevant to the department, and can be used to populate the system with a list of teachers, timetables, etc. In addition, a Google Earth plugin and the exhibition of three-dimensional content would make the system more engaging, as well as providing interesting methods for interaction with a mobile device.

The system will include two applications, a desktop application, which will be developed in .Net, for its performance, security, and included features, while the mobile application will be developed in Android, using Java, since this is the standard approach.

4 Development

In this chapter, both the mobile and the desktop application will be described in detail. In addition, some implementation details will also be presented.

4.1 Introduction

As mentioned, DETI interact is a system that uses two different applications: a mobile application, running on an Android device, and a desktop application running on a computer, which displays information on a large screen.

The Android application is tasked with collecting different gestures performed by the user. With no information being displayed there, the user can focus on the large screen, and use the mobile device merely as another peripheral.

The desktop application is responsible for displaying the information, as well as interpreting the gestures from the mobile application, and responding accordingly. This required a series of libraries to be developed. These were needed for fetching and processing the data from the DSD server, to enable the inclusion of and XNA application, and to interpret the gestures performed by the user.

4.2 Android Application

To interact with the desktop application a mobile application running on the Android operating system was developed. This application connects to a DETI Interact computer and transmits data through a Bluetooth connection. A Bluetooth service was developed to perform all Bluetooth-

related operations. In addition, the Android SDK supplies a `GestureDetector` class that was used for the detection of gestures. The values from the digital compass are also sent via Bluetooth to the computer.

4.2.1 Bluetooth Service

The `BluetoothService` class controls the communication with the desktop computer. To perform the connection, the Bluetooth MAC address of the computer must be known. This information is gathered when the devices are paired (as stated in 4.3.2.3, Android's Bluetooth API enforces device pairing). When the application is launched, a list of connectable Bluetooth devices is displayed. Once the user chooses a device, its address is passed to the `BluetoothService` class, which then tries to connect to it. If the connection succeeds, the input and output streams are initialized, enabling the application to read and write from them.

A `write()` method is supplied by this service that can be used to place an array of data into the output stream of the Bluetooth connection. The streams between two Bluetooth (or Serial) endpoints can be written or read at any time, by different clients. If two clients were to write at the same time, whoever reads the stream afterwards will find a mixture of both messages. In order to avoid this situation, the provided `write()` method only allows one message to be written at any time. Whenever the system needs to write some data to the stream, the `write()` method blocks any further calls, in order to avoid conflicts. While the method is blocked, any awaiting messages are discarded, since the system is expected to respond to user input in real time, and answering these messages would result in the system responding with some lag.

4.2.2 Main Application

The main activity for the Android application requests a Bluetooth connection to the aforementioned service. The information collected by the sensors and sent to the Bluetooth services is placed in a string, with a predetermined size and structure. This structure, shown below, identifies the gesture, as well as three distinct fields, for the X, Y, and Z components of a scroll or fling, or the Yaw, Pitch and Roll components from the device's orientation. Unused fields are left with a null value.

Bytes:	0	1-2	3	4-12	13	14-22	23	24-32	33
Content:	"["	Gesture ID	":"	Value 1	":"	Value 2	":"	Value 3	"]"

This structure has a fixed size to aid the computer in the decoding. This eliminates some problems that existed on an initial version of the system where the computer was decoding messages at a slower rate than they were being sent. This caused some conflicts in the messages such as messages cut in half, or concatenated messages, which interfered with the gesture detection.

Concerning the detection of gestures on the device's touchscreen, the `GestureDetector` class offered by Android's API identifies most of the needed gestures such as Tap, Scroll, and Fling. However, the Pinch gesture, used to zoom in and out, is unavailable in the API for Android 2.1. A `ScaleGestureDetector` class was introduced in version 2.2 of the API to allow the detection of pinch to zoom gesture.

This constitutes one of the main problems of the Android platform: fragmentation. Mobile operators and OEMs (Original Equipment Manufacturers) load devices with their software, or modified firmware versions, which have to be adapted when Google launches a platform update. New features, such as the `GestureScaleDetector`, are added to these updates, and are not backwards compatible. Most devices will only receive these updates later if at all. The result is that the Android ecosystem includes devices running android versions from 1.6 up to 3.0. If an application uses a feature from the API from version 2.2, every device running Android 2.1 or under will not be able to run the application. For this reason, and since development was started when Android 2.1 was the most recent Android version, the zoom gesture had to be implemented from scratch. This is done by detecting two fingers on the device's touchscreen, and accompanying the variation in spacing between the fingers.

4.2.2.1 Supported Gestures

The set of gestures currently supported by the system include Scroll, Fling, Tap, Long Press, Rotation, and Zoom.

Scroll: scrolling (Figure 4.1) is done on the mobile device by touching a point on the device's touchscreen, and dragging the finger to another position. Any direction on the XY axis is valid and therefore transmitted to the desktop application.

Fling: the fling gesture (Figure 4.2) is a quick finger swipe, across the device's screen, in any direction. It is usually used to perform an accelerated scrolling gesture, known as Kinetic Scrolling.

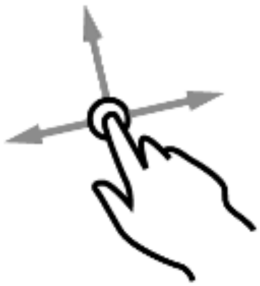

Figure 4.1 – The Scroll Gesture (Ideum 2011).

Figure 4.2 – The Fling Gesture (Ideum 2011).

Figure 4.3 – The Tap Gesture(Ideum 2011)

Tap: a tap (Figure 4.3) consists of touching the screen for a short duration of time. It is usually used to perform selections.

Long Press: a long press (Figure 4.4) is done by touching the screen for a longer period of time (about one second), which differentiates itself from the tap gesture, as well as accidental touches. This gesture is usually used to trigger the display of a context menu.

Pinch to zoom: zooming (Figure 4.5) is done placing two fingers on the touchscreen and moving them closer together or apart, to zoom out or in, respectively.

Figure 4.4 – The Long Press gesture (Ideum 2011).

Figure 4.5 – The Pinch to zoom gesture (Ideum 2011).

Figure 4.6 – The Rotation gesture.

Rotation: the values from the device's digital compass are constantly polled and sent to the desk-top application. These values are used to detect several rotation gestures (Figure 4.6).

The development of the desktop application can be targeted to take advantage of this gesture library.

4.3 Desktop Application

As discussed in section 3.2.1, the Web Services available in the DSD platform, albeit limited, are enough to list lecturers and generate schedules. These two features were chosen to integrate the first version of the application. In addition, since current mobile devices are equipped with multi-touch displays, accelerometers and digital compasses, a map and a three-dimensional object viewer were also included in the application.

During application development, attention was taken to ensure system extensibility, so that it would be easy to add, remove, or modify core features.

4.3.1 Structure

The desktop application was developed in C# using Windows Presentation Foundation (WPF) for the user interface, using the Model-View-ViewModel pattern where appropriate. This prompted for the structure shown in Figure 4.7.

The application presents a user interface with a series of pages, with different content being presented on each page. In the initial version of the application, there is a page for the list of teachers, a page for the timetables, a page for the Google Earth plugin, and a page for the 3D model viewer. There is also an additional page where help is presented to the user. Every page hosts a WPF control, which is tasked with displaying the specific content, and offers several interaction possibilities. This is achieved by using the developed `DetiControl`. This control provides a series of properties, methods, and events so that the application can control its operation. By adding a `DetiControl` to the application, a new page is automatically created on the UI and it will operate in the same way as any existing control.

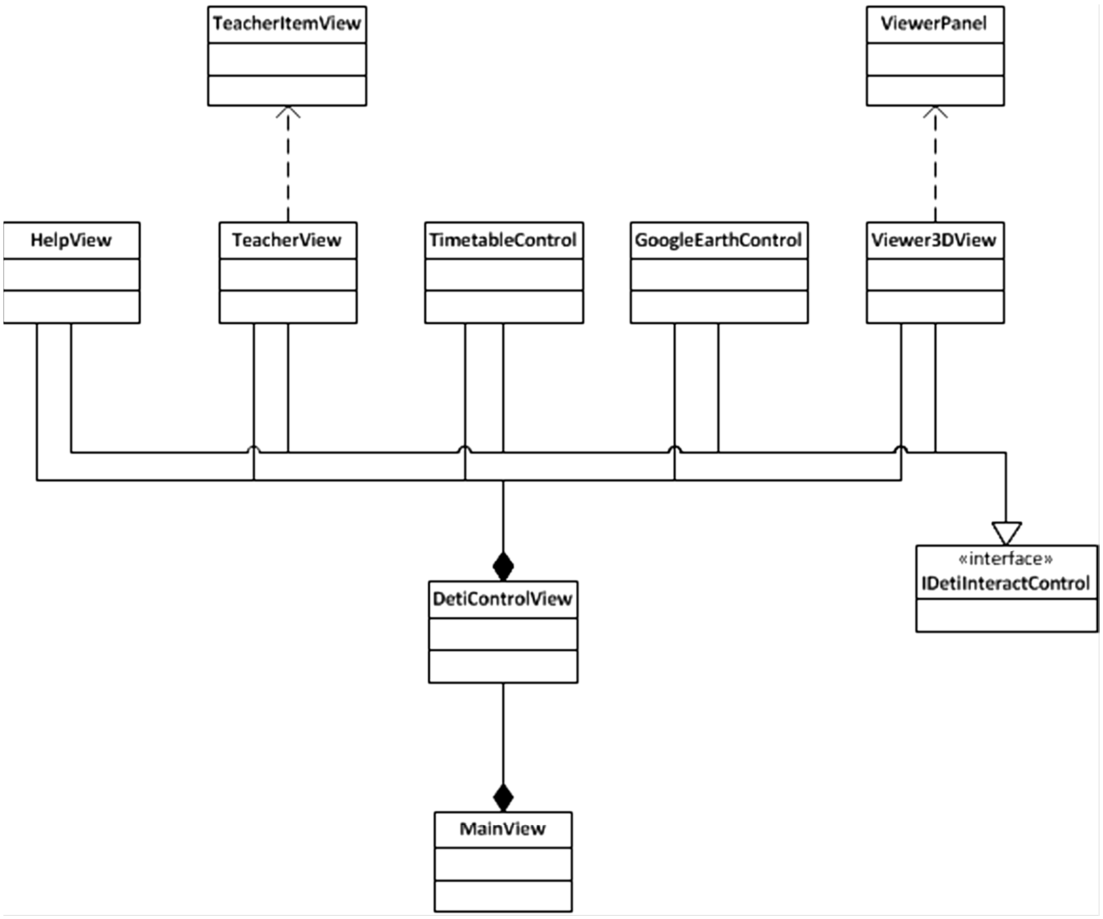

Figure 4.7 – Structure of the main application.

Each page must also display some content. In order to do so, the `IDetiInteractControl` interface is supplied. This interface provides methods that allow the system to perform operations such as triggering demonstration mode, and reacting to user input. Using these two components, all the developer has to do to add new content to the application is:

- Create a new WPF control to host the desired content.
- Implement the `IDetiInteractControl` interface. The control should have its own animation routine, and should respond to user input.
- On the main UI file (`MainViewModel`) used the provided method to add the control, and the title to be displayed on the page:
    - `AddWindowControl(new DetiControlView(new WPFControl(), "<Control Name>"));`

The developed control will have been added to the user interface, a new page will have been created, and the control will respond to user input and animation events.

4.3.2 Developed libraries

In order to offer the features discussed in Chapter 3, a set of auxiliary libraries was developed to provide the necessary content, namely, the DSD Provider library, and the Model Viewer XNA library. The DSD Provider communicates with the DSD server to collect the necessary information to present the lecturer list and the timetables, and the Model Viewer XNA library consists of an XNA application, which will render the models. Including a Google Earth plugin does not require the use of an auxiliary library, so it can be done within the main application.

In addition, two separate libraries were also developed, one that allows interaction from remote devices, the Control library, and one used to perform logging of events, the Logger library.

4.3.2.1 DSD Provider

As discussed in Chapter 3.2, DETI Interact needs data that is stored on the DSD platform, which is accessible via web services. Accessing web services, however, can take some time due to communication delays. Furthermore, processing the vast amount of data returned can also be time-consuming. These reasons prompted for the creation of the `DSDProvider` class library (Figure 4.8).

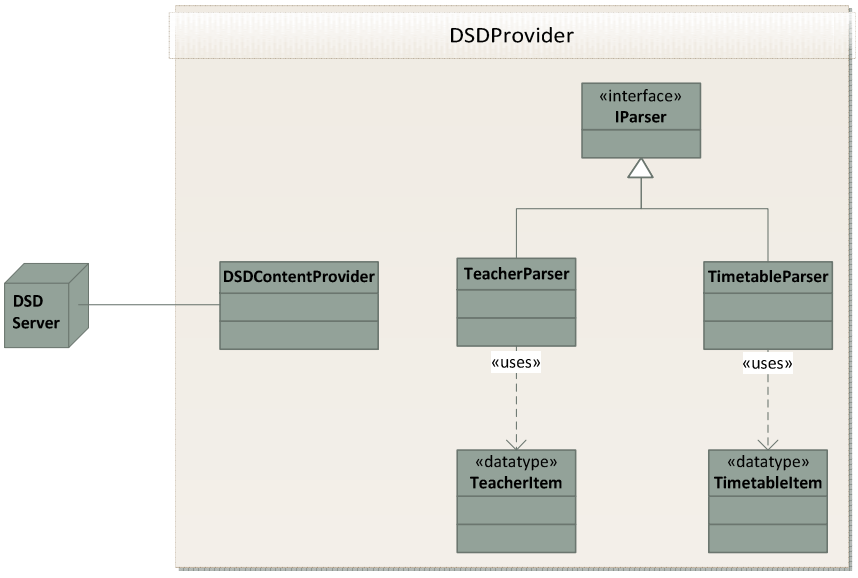

Figure 4.8 – The `DetiInteract.DSDProvider` class library.

This library is composed of two parts: the `DSDContentProvider` class, tasked with collecting all the data, and the Parsers, that work on the collected data.

4.3.2.1.1 DSDContentProvider

The `DSDContentProvider` class is responsible for accessing the Web Services and retrieving information from the server.

In order to use a Web service in a .NET project a proxy class must be created and compiled, this class is generated from a reference to the Web Service, typically the WSDL file. Using Visual Studio, this is achieved by adding a Service Reference to the project and assigning the address to the web services. Once the reference is added, the proxy class is generated automatically. The method for consuming a web service would be similar to that of calling a method from an external library.

4.3.2.1.2 Parsers

The Parser classes are functional components that work with the data provided by the `DSDContentProvider` class. Each parser processes a specific set of data and sends the result back to the application.

To ease system extensibility, an `IParser` interface was created. This interface lists the methods that every parser should implement. This allows the developer to determine how the parser will perform its functions and what data will be processed. The interface also publishes an event that can be used to notify the application of changes in the parser (such as when new data is available).

In the present state of the DETI Interact system, two distinct parsers are implemented: the `TeacherParser` and the `TimetableParser`.

In terms of performance, since calling Web services usually takes up more time than calling an internal operation, `BackgroundWorker` instances were created in each parser to collect and process the information. A `BackgroundWorker` allows an operation to be run on a separate, dedicated thread. Time-consuming operations which the parsers are likely to perform can make it seem as though the UI has stopped responding for a period of time, running it on a separate thread maintains responsiveness, improving user experience.

TeacherParser

The system that was previously running in the lobby contained a web page that displayed information regarding all the teachers of the department. The page would automatically scroll to different positions so that all would be shown. Information about each teacher was organized in a business card fashion with his or her name, photo, web page, office door number, email, and phone

number. This behaviour was to be reproduced in DETI Interact so this information had to be extracted from the list of teachers that the `DSDContentProvider` had fetched.

Extracting the required data from the `DSDContentProvider` is a simple operation although it requires looping through all the elements in the list. The `BackgroundWorker` loops through the list and extracts a teacher record, it then sends this record the `ProgressChanged` handler, which collects the required data and places it in a data structure. This structure is then sent to the UI thread via the existing event.

TimetableParser

The web page for the DSD platform (http://dsd.av.it.pt) allows the user to check timetables for each course and grade at the department, this feature was chosen to integrate the initial version of the DETI Interact system.

Creating a visual representation of a timetable requires a more careful approach since the Web services can only return textual information such as a list of all the subjects, classes, and classrooms, class times, etc., which constitutes an immense amount of data to be processed. In order to determine when a certain class takes place, it is necessary to navigate through six tables, which can be an arduous task for the CPU. Since the UI thread has to be able to quickly draw a timetable, most of the work will have to be done in the parser.

After accessing all the lists in the `DSDContentProvider`, the parser must extract the required data in order to create every item for the timetables. This data will be placed into a data structure (the `TimetableItem` class) that will be sent to the UI thread for drawing. The classic approach of looping through all the lists attempting to match IDs would result in an algorithm with a complexity of $O(N^6)$. This is where LINQ becomes a very useful asset. The querying capabilities of LINQ enable lists to be joined making it only necessary to loop through the resulting list once, lowering to complexity of the algorithm down to O(6N). This resulting list contains all the `TimetableItem` instances for a given year and course combination, and these can now be sent to the UI thread via the existing event.

4.3.2.2 XNA Model Viewer

One of the features previously discussed had the application display three-dimensional content. This would be used as a virtual tour of the department, allowing the user to visualize models or navigate in a virtual representation of the department.

There are numerous ways to display three-dimensional content on a computer, including OpenGL, DirectX, VTK, XNA, and WPF. Ideally, this 3D content would be rendered within the application instead of a standalone application. For this scenario, the first choice would be to use the 3D API included in WPF, as it would render content directly in the application window. However, this API is not as full-featured as OpenGL, DirectX, or XNA, meaning it would not scale as well with larger, more detailed scenes. The alternative APIs render 3D content to a specific buffer, such as the full screen buffer, or a given window handle, so it is required to replace this target buffer with a certain portion of the window.

Previously, rendering 3D content within an application required a Windows Forms control, which could be hosted within a WPF application. While this worked without problems, performance was severely affected. With version 3.5 of WPF the D3DImage `ImageSource` was introduced, which can be used as a render target for DirectX. Since XNA is a managed library built on top of DirectX, this was the library of choice for the Model Viewer.

The Model Viewer library is a standard XNA library project. It includes a series of 3D Models in FBX (Autodesk) or X (DirectX) formats, which are loaded and rendered by request of the UI thread. It also provides basic camera interaction.

4.3.2.3 Control

One of the main objectives of the DETI Interact system is the ability to control it via an Android mobile device. This is possible through the `DetiInteract`.Control library. This library is used to receive data from the Android device and interpret a certain gesture; it will then notify the UI thread of the gesture so that the application may perform accordingly. This implementation allows the application to decide how to deal with each gesture; it also allows the library to be reused in a different scenario.

This library contains a Controller class, and the `CommHandler` classes. In this version of the system, the chosen communication interface was Bluetooth, resulting in the creation of the `Bluetooth-`

`CommHandler`. Other `CommHandler` classes can be added as required, ensuring reusability and extensibility of the library.

4.3.2.3.1 BluetoothCommHandler

The `BluetoothCommHandler` class manages communication between Bluetooth enabled mobile devices. While the system is targeting Android devices in particular, any Bluetooth device will work with this class. All Bluetooth communication is handled by In The Hand's 32Feet.NET library (Foot). This library creates a higher level of abstraction on top of the Win32 API.

Setting up communication between Bluetooth devices is a usually straightforward task but Android's Bluetooth API requires those devices to be paired. Since the system is to be deployed to computers connected to a large display, with input methods inaccessible to the user, pairing must be done automatically and seamlessly. This presented a problem during development since different versions of Windows supported different Bluetooth specifications. Windows Vista and Windows 7 both support Bluetooth v2.1, which automatically performs Simple Secure Pairing with devices that support it. With Simple Secure Pairing, both devices agree on a sequence of characters, which the user then has to confirm on both devices. This feature overrode any attempt to pair the devices within the application, requiring user input on the Operating System level.

Windows XP did not present this problem, other solutions would include using and older Bluetooth adapter on the Windows machine, or using the Win32 API directly. The Android SDK version 2.3.3 features an API for insecure Bluetooth socket connections (CellPhonesMarket 2011), which will avoid this issue

To perform device pairing within the application a callback must be set to answer pairing requests. This callback only has to define the pin that the mobile devices must use to pair. This way pairing will be done and the Operating System will not interfere in the process.

Communication between devices using Bluetooth is similar to communication using a Serial interface (Bluetooth is essentially a wireless implementation of Serial communication). The devices must know the MAC address for each other, open a socket, and write into or read from that socket. Since the DETI Interact system will be deployed in a public setting, many Bluetooth-enabled devices will be in range, and polling every device would not be a very practical approach, instead the system will be constantly waiting for a device to request a connection. After accepting the connection, the system asks the device for the shared Stream. This stream will then be used to read the packets

sent from the mobile device that contain data pertaining to the gestures performed on the device. Since waiting for a connection request from a device is a blocking operation, and maintaining a continuous connection with a device is time consuming, the code must be run on a separate thread.

The data read from the stream encodes data from the mobile device. This data consists of a gesture identifier and a set of values corresponding to the gesture (for example, the X and Y speed of a fling); however it can also notify when a device is about to disconnect. This resets the listening process, leaving the system to wait for a new connection. Gestures are reported to the Controller via an event.

4.3.2.3.2 Controller

The `BluetoothCommHandler` class handles communication with remote devices via a Bluetooth connection and then report to the Controller class. Additional classes can be added to this library as needed to allow for different methods of communication. The `Controller` class' purpose is to manage the existing `CommHandler` classes. This class interprets that data received by the `CommHandlers` and notifies the UI thread via an event with details regarding to gestures performed on the mobile device.

4.3.3 Displayed Content

Various controls were included in the initial version of DETI Interact, namely a control that displays a list of teachers, a control that displays the students' timetables, a control that allows navigation in an interactive map, and a control that enables the user to manipulate a three-dimensional object. An additional page was added to the application displaying helpful information to the user regarding system usage.

As specified in the previous chapter, the displayed content will be organized in several pages, in a tabbed interface, with vertically oriented tabs arranged along the edges of the screen. Given the available set of gestures, it was decided that the touch-based gestures would be used to interact with the displayed content, while the orientation-based gestures would be uses to alternate between pages. The reason behind this choice was that the use of the digital compass would place the user under a higher cognitive load, so it would be more appropriate to leave it to secondary actions. This choice was later confirmed with user testing, which can be seen in Chapter 5. Interac-

tion with the content displayed in the pages is implemented by each page, which will be detailed below.

4.3.3.1 Teacher Page

As stated in 3.1, the previous system displayed a list of information regarding the teachers of the department. Information about each teacher was organized in a business card fashion and displayed in a matrix on a web page. This page would automatically scroll to pre-set locations so that all teachers would be displayed. The teachers' page in the new system implements a similar behaviour.

The teacher page consists of a Teacher Control, which hosts a list of `TeacherItemControls`. Both the controls are built using the MVVM pattern, taking advantage of the databinding capabilities of WPF. The `TeacherItemControl` was developed to house the information for each teacher. The ViewModel for the `TeacherControl` is responsible for dealing with the data. It starts by creating an instance of the `TeacherParser`, which returns a list of `TeacherItem` instances. These are used to create the `TeacherItemView` controls, which are placed into an `ObservableCollection`. WPFs databinding engine ensures that the contents of the listbox on the page match the contents of the collection at all times.

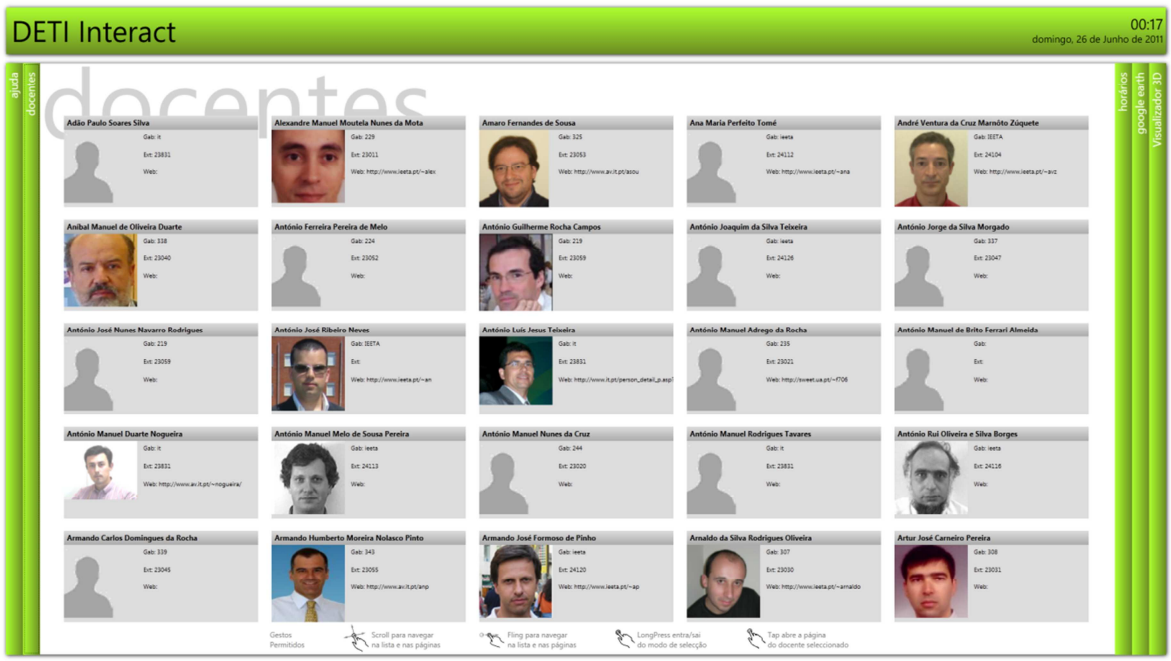

Figure 4.9 – The Teachers' page.

This control is configured to use all the available screen space, regardless of resolution, meaning that more information will be visible in the larger screens whilst remaining easy to read. For instance, a 1080p resolution screen will display five columns of teachers, while a 720p resolution screen will only display three columns. Figure 4.9 shows the final look for the teachers' page.

In addition to displaying the list of teachers, this control also displays the web page of any selected teacher if the URL is in the database. For the page to be displayed, a Web Browser control is overlaid on the list of teachers (Figure 4.10). When selecting a teacher, the address to his webpage is passed to the browser and the page is loaded. The Web Browser control was configured to prevent new windows from being opened, which would cause the application to lose focus, and become concealed by the newly opened window.

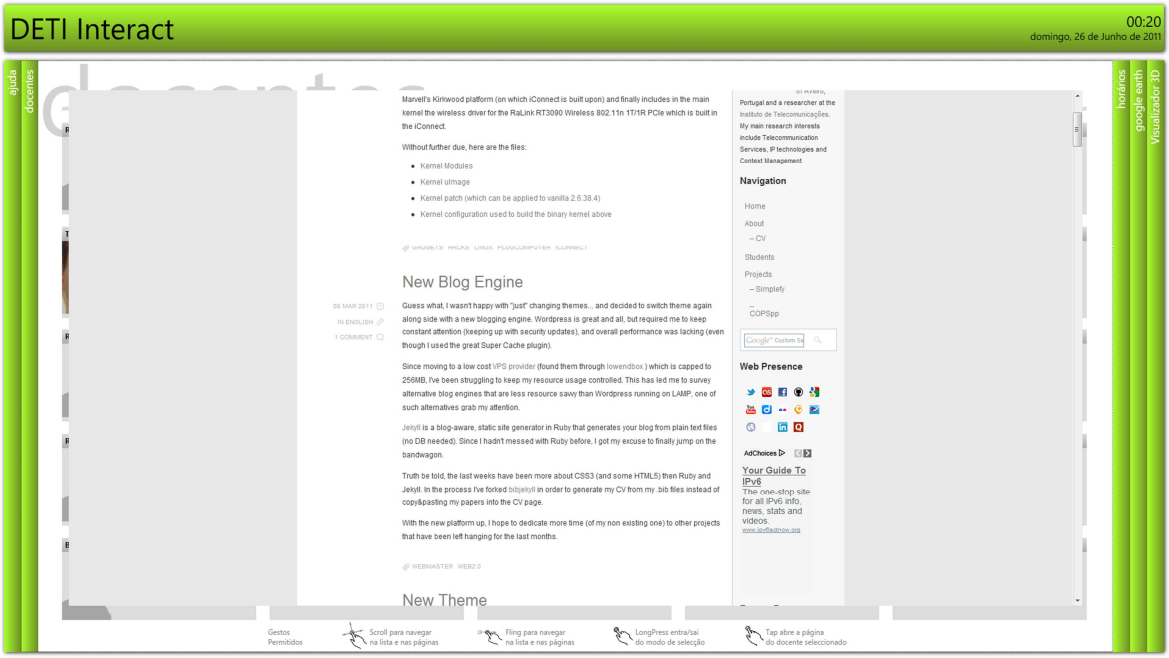

Figure 4.10 – The Web browser, visible after a user selects a teacher.

4.3.3.1.1 Interaction

To interact with this control, several gestures are provided. The users can scroll vertically along the list of teachers, or use a vertical fling gesture to trigger kinetic scrolling. In order to browse the teacher's webpage, the user must enter selection mode. This is done via a `LongPress` gesture. This gesture is used to enter and exit selection mode. Upon entering, the teacher closest to the centre of the page is automatically highlighted. The user can then use flings, in any direction, to select any

adjacent teacher. Once the desired teacher is selected, a simple Tap gesture will launch the web page.

Once the Web Browser control is visible and the webpage loaded, the user can scroll or fling, in any direction, to view the full extent of the webpage, in a manner similar to what is done in modern mobile web browsers. To leave the browser and return to the teacher list, the Long Press gesture is used once again.

4.3.3.2 Timetable Control

The previous chapter specified how the `TeacherParser` was used in creating the teachers' page; a similar approach was taken to develop the timetables' page, using the `TimetableParser`. The parser was built in a way that frees up the control of the greatest part of the work; however, the control still has to place every item in its correct place, in order to create a readable timetable.

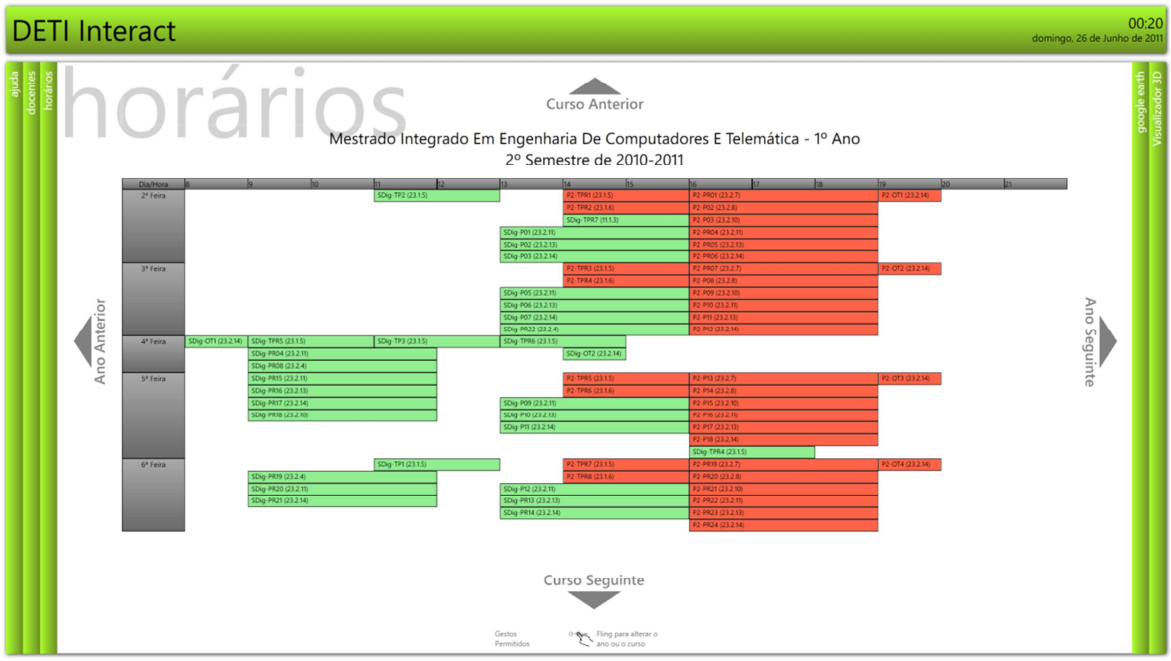

Figure 4.11 – The Timetables' page.

Analogous to the teacher control, an instance of the parser is created and every time the Changed event is triggered, the control draws to the screen. Data is passed to the control via a specific data structure containing subject name, classroom number, start time, duration, etc., called a `TimetableItem`. The timetable is represented by a Grid control where the rows represent days of the week, and the columns represent time. Before any work is done, the grid is configured for the appropriate degree program and year. The classes are then positioned on the grid, on their correct

places, colour-coded by course, so that they can be easily identified. In addition, since many classes can occur at the same, the grid's rows expand to accommodate them. The resulting timetable page can be seen in Figure 4.11.

4.3.3.2.1 Interaction

Interacting with the timetables via the Android device is limited to the Fling gesture. The user must swipe horizontally on the screen to navigate through the different years, or swipe vertically to change the degree program. Every time the user switches to a different timetable, the parser re-starts and process is repeated.

4.3.3.3 Google Earth Control

Another feature chosen to integrate the system is the inclusion of a Google Earth plugin in the application as it can also be used to validate the use of a mobile device for interaction with large displays, since it can offer a use for the touchscreen similar to what is already available on mobile devices and other touchscreen-equipped computers. Google Earth is presented on a page (Figure 4.12), and the user uses his/her phone to pan, tilt, or zoom.

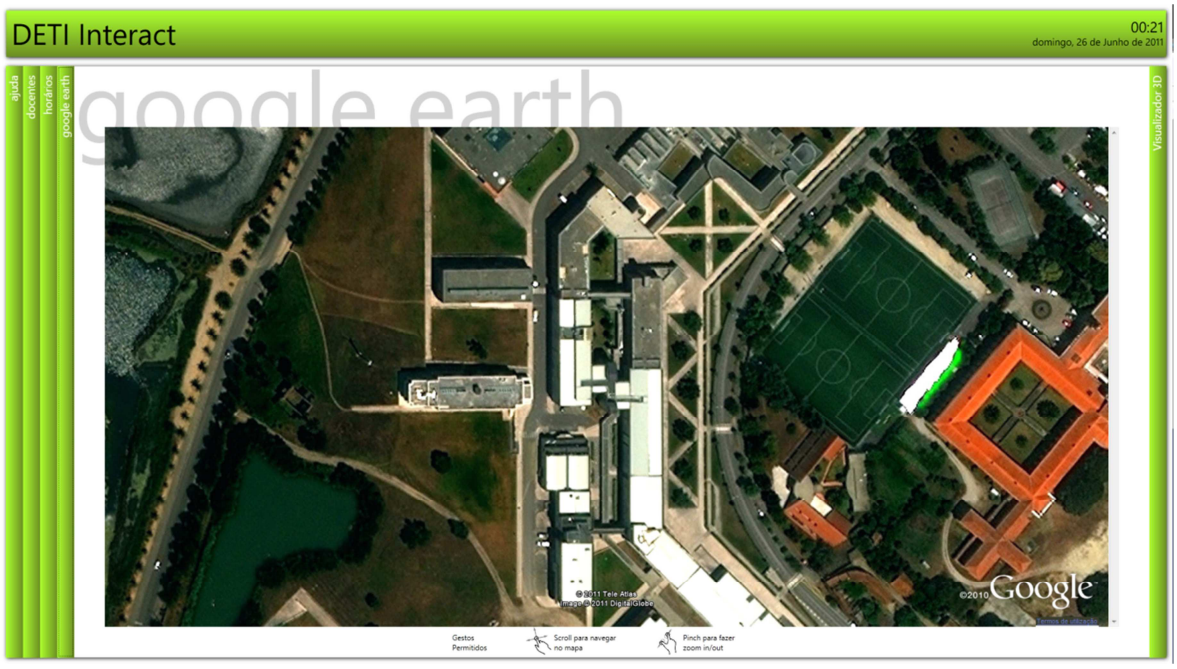

Figure 4.12 – The Google Earth page.

Google Earth works as a plugin integrated into a web browser. For developers, Google offers a Javascript API that allows web applications to host a Google Earth control. DETI Interact, however, is not a web application, meaning that there is no direct way to run the plugin in a native application.

In order to overcome this difficulty the developed control would have to be able to access the Javascript API from within the application. This was achieved in two steps. Firstly, it was necessary to find a way to draw a map on the screen, and allow interaction with it (panning, zooming, etc.). This was done via an HTML file. This file uses Google Earth's Javascript API to display the Google Earth control and works as a regular web page: it can be opened in a web browser, in which the Google Earth plugin is loaded and navigation within the plugin and interaction with it is as expected on any other page. The file also provides a small selection of Javascript functions that can be used to manipulate the map.

The second step consists in finding a way to render the HTML file inside the application, and using the provided functions to interact with the map. This is done by loading the file into a Web Browser control. This control renders the HTML content like any conventional web browser application, and using the provided `InvokeScript()` method, it is possible to call Javascript functions from within the HTML file. This resulted in the creation of new control to host the web browser

4.3.3.3.1 Interaction

Interaction with the map from the Android device is done as if the map had been drawn on the device's screen. Moving a finger in any direction will pan the map, and using the familiar pinch gesture, will zoom in or out, centred on the current position.

4.3.3.4 3D Viewer Control

The 3D Viewer control presents the content rendered by the XNA project mentioned in 4.3.2.2 within the WPF application. The 3D Viewer control is divided in two components: a panel to host the aforementioned D3DImage control and perform interoperation with XNA and DirectX, and a control to host this panel, which is tasked with animation and user interaction.

Viewer 3D Panel

The Viewer3DPanel hosts the D3DImage control and sets it up as the render target for the Model Viewer Library. This cannot be done directly although it is planned for future releases of both WPF and Silverlight (Microsoft 2011). This, however, can be achieved by using P/Invoke and Reflection to access specific DirectX and XNA content (Morril 2008).

In order to render a 3D scene into an image container inside the application, the render target of the XNA application must be switched to this container. This is done using the Interop Services

provided by the .NET platform. These services allow a developer to access features of native APIs, unavailable on the .NET framework, such as the Win32 API and DirectX. In the Viewer3D panel, P/Invoke is used to fetch the `IDirect3DTexture9` and IDirectDevice9 interfaces from the DirectX API. In addition, Reflection is used to get the private `ChangeDevice()` and `Initialize()` methods from the XNA API.

The Viewer3D Panel control can now access the `ModelViewer` library discussed in 4.3.2.2, and set the render target to the image container it hosts.

Viewer 3D

The Viewer3D is a control that wraps around the previously created panel. It was developed using the MVVM pattern. Its View Model is databound to the mentioned XNA application, while the view is responsible for animation and user input handling.

The Viewer3D control displays a list of all the three-dimensional models available in the system. It supports a selection mode similar to the one developed for the Teacher page. Once one of the models is selected, the Viewer3D Panel is displayed and interaction with the model is possible (Figure 4.13).

Figure 4.13 – The 3D Viewer Control, displaying a model synchronized with the mobile device's digital compass.

4.3.3.4.1 Interaction

Interaction with the Viewer page is similar to the interaction developed for the Teacher page. This page displays a list of the available models, in which the user can scroll of fling to browse its contents. A Long Press enters and exits a selection mode that allows the user to load a desired model via a Tap on the device's touchscreen. When a model is loaded, the Viewer 3D Panel is displayed, and the user can use his phone to manipulate the model. Orientation changes on the phone will directly affect the model, as if the user was holding the displayed model in his/her hands.

4.4 Deployment

In order to get the system working, the desktop application had to be installed and the mobile application had to be installed on the phone. The installation of the desktop application faced a series of problems. Since the application heavily depends on external sources (the Web Services from the DSD platform) and some features from the .Net framework, it was not possible to simply run the binary on the target machine. The application had to be installed in order for all the dependencies to be resolved. This was done using the provided ClickOnce functionality. This feature generates an executable binary for the application along with the needed assemblies and a configuration file. When the binary is run, the application is automatically installed and run.

Once the application was installed, it was also necessary to configure the machine so that it would run unattended. This involved setting up an automatic logon, configuring the anti-virus software and configuring the computer's software and services that run automatically during boot, so that DETI Interact can run without interruptions.

4.5 Conclusion

In Chapter 3, the system architecture was detailed, along with the two applications that would make up the system. The mobile device would work simply as a peripheral; therefore, the developed mobile application simply gathers data from the sensors and sends it via a Bluetooth connection. The desktop application then processes this data and interprets the gestures. The main function of the desktop application is to present content to the user. It connects to the required services and populates the screen. The system was developed and structured in a way that simplifies future modifications and extensions, so that additional content and interaction methods can be introduced to the system.

5 User Evaluation

In this chapter, the evaluation of the system will be discussed. In addition to user testing of the system, an application to evaluate Fitts' Law was developed to compare the use of a mouse versus the use of a mobile device as a means of interaction, in a series of selection tasks.

5.1 User Testing

An important stage in the development of an interactive system lies in user testing. Since the users will be the primary clients of the system, it must be ensured that they understand how to operate the system and enjoy using it. These tests assist in identifying errors in the system and problems in the user interface, which can then be addressed in order to deploy a fully functional version of the system.

DETI Interact proposes the use of a smartphone to control the display of information on a large screen. This was also studied during the development of DetiGuide (Palha 2010), the results, however, were inconclusive, consequently, with DETI Interact instead of proceeding straight to test the developed system, it was decided to conduct a preliminary testing session to evaluate Fitts' Law. The model developed by Paul Fitts relates the size of a target and its distance to the mouse cursor with the time it takes to select it. This would enable testing different usage scenarios where a smartphone could be used to control a computer. In addition to this test, a conventional test to evaluate the usability of the system was also organized.

5.1.1 Fitts' Law

In order to compare the ease of use of any given system using a mouse or a mobile device, an application was developed to evaluate Fitts' Law. Fitts proposed a formal relationship that models speed-accuracy trade-offs in rapid, aimed movement. This model predicts that the time required for a user to move to a target area is related to the distance and size of the target. It has been formulated mathematically in several different ways; the following equation, known as the Shannon formulation is preferred since it always provides a positive rating for the index of difficulty for each task (MacKenzie and Buxton 1992).

$$MT = a + b \log_2\left(\frac{2A}{W} + 1\right)$$

In the equation, **MT** represents the movement time to hit the given target, with **A** and **W** representing the distance to the target (amplitude), and the target's width, respectively. Constants **a**, and **b** are determined empirically with user testing. Once the results are processed, **a** expresses the start/stop time of the devices, and **b** represents the inherent speed of the device. The term $\mathbf{\log_2\left(\frac{2A}{W} + 1\right)}$ represents the index of difficulty, **ID**. In addition, $\mathbf{1/b}$ is called the index of performance, **IP**, of a device.

To compare the performance between the use of a mouse and mobile device, an additional application was developed, using .NET and WPF. This allowed for the reuse of components such as the Control library (described in chapter 4.3.2.3), which gives access to the Bluetooth stack as well as the gesture handlers. This application generates circular objects on de screen, which the user must click. The size and position of each circle on the screen is random, in order to randomize the index of difficulty. To ensure a wide range of data, a total of 50 targets are generated. There were five different circle sizes used on the application: 20px, 40px, 80pc, 120px, and 240px in diameter. The smaller sizes, 20px and 40px, were displayed with roughly 10mm and 20mm, which are confortable sizes to be used for finger-based interaction on a touchscreen (Microsoft 2011).

There were three different interaction methods available:

- The mouse, to be used as a reference for the other tests – the mouse is used to move the cursor, and the mouse buttons used to click on the targets;

- The touchscreen on the mobile device – the touchscreen is used to move the mouse cursor as a computer touchpad would, and tapping the touchscreen would send a click action to the application;

- The digital compass on the mobile device – device orientation manipulates the mouse cursor like a joystick, and tapping the screen clicks the targets.

Given the random nature of the tests, each user can perform the three tests. Since both mobile interaction methods are slower than the computer mouse, the mouse's sensitivity was reduced, to approximate the speed at which the others operate. All users were positioned at the same distance from the display, delimited by a table that would provide a surface for the mouse. Users remained standing throughout the tests.

5.1.1.1 Results

The tests were performed with 34 computer engineering students, resulting in 1700 results per test. All the tests were completed successfully, with the clear worst interaction being the use of the digital compass. The main problems with both methods of interaction on the mobile device were directly related to the accuracy in the collected values: the touchscreen has a much lower resolution than the computer mouse, and the digital compass, even in its fastest polling state, was too slow to manipulate the cursor comfortably.

In general, both methods have an overall worse performance when compared with the computer mouse. During the test, it was observed that when tapping the touchscreen users would often slightly move the cursor. This affected both interaction methods. The users felt that the smaller targets (roughly 1cm in diameter) were the hardest to click. In this case, both methods were distressing since very slight adjustments on the touchscreen were often detected as a Long Press by the Android OS, and the sensibility of the compass would have the users circling around each target without being able to position the cursor correctly. However, using the compass on these smaller targets still offered better precision. Apart from being more efficient with the touchscreen, some users enjoyed using the compass, since they could tap on the screen while tilting the device. In the end however, the preferred method of interaction was the touchscreen.

The results of this series of tests can be seen plotted in Figure 5.1. This graph shows the time it took for each user to select each target, using the three interaction methods. The horizontal axis expresses the index of difficulty, which relates the size of a target with the distance from the cursor.

The mouse interaction, used as a baseline, offers the best performance as it can be seen in the chart. Although the surface on which the mouse was placed was not optimal, which can be seen in some of the values, users were able to click the targets quickly. The interaction methods using the digital compass or the touchscreen on the mobile device show a worse performance, with the digital compass displaying the worst results.

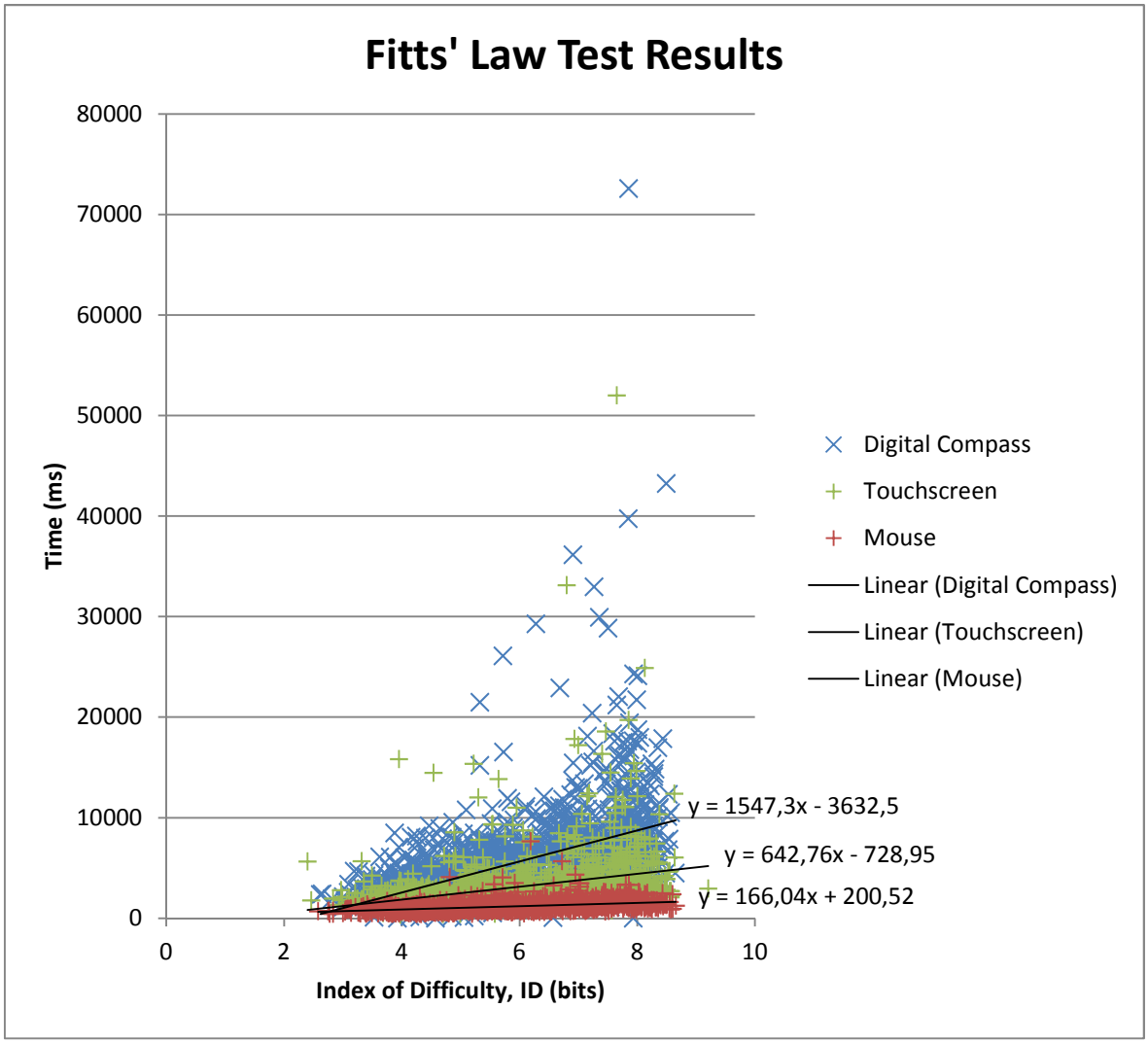

Figure 5.1 – Plot of the results obtained in the Fitts' Law test, along with a regression line for each method.

In addition to evaluating the time it took for the users to click on a target of a certain difficulty, it was also interesting to relate the elapsed time with the size of the targets. Figure 5.2 shows how long it took the users to click on each target, given its size. As specified previously, the size of the targets ranged from 20px to 240px. The graph shows that, as anticipated, users spent more time in attempts to click the smaller targets, with the larger targets earning the better results. This shows

that when creating user interfaces for this type of interaction, buttons, links, or other controls should have a considerable size, in order to provide the user with a better experience.

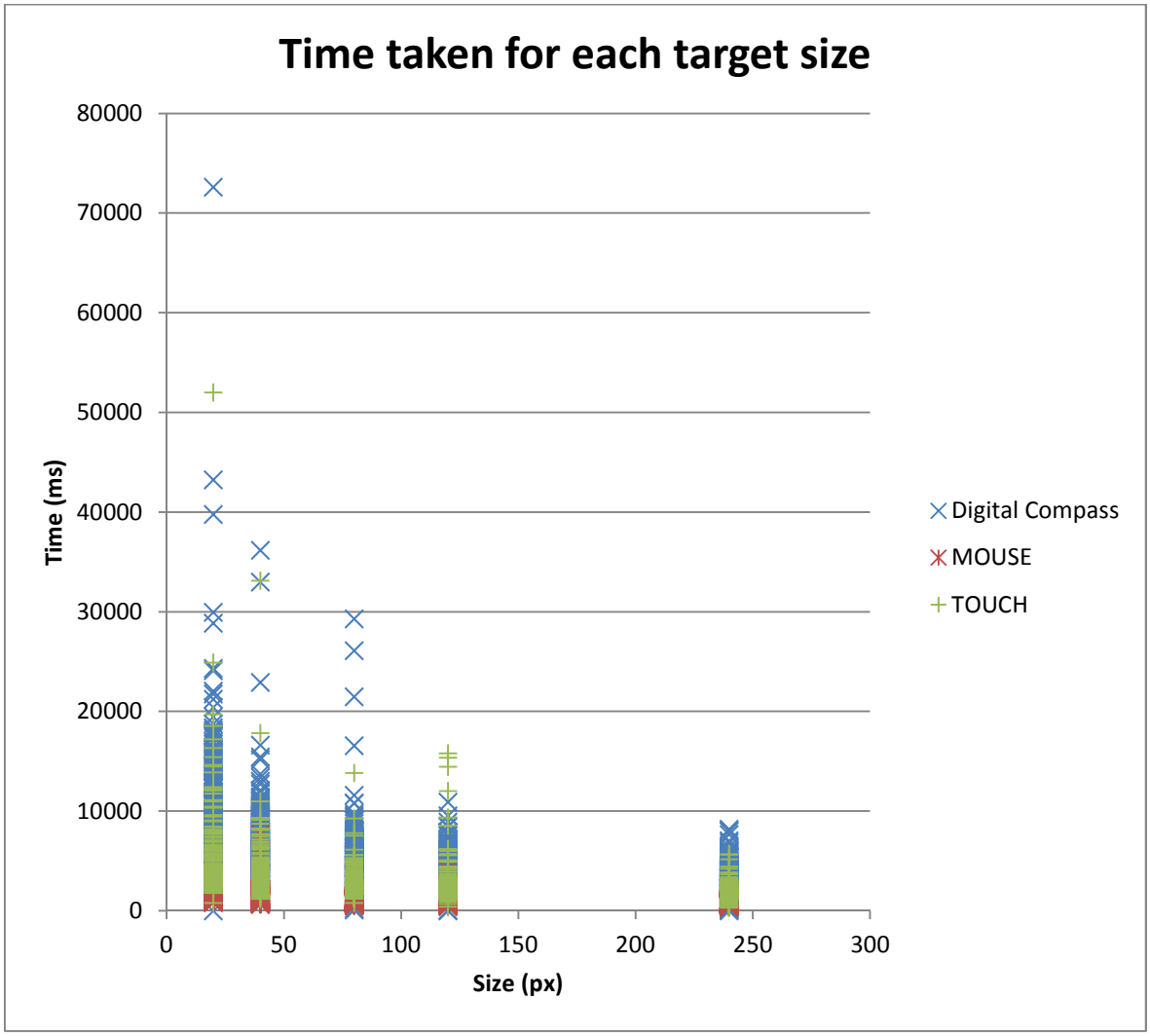

Figure 5.2 – Plot that relates the time taken to select each target with its size.

5.1.2 DETI Interact

The tests based on Fitts' Law assist in evaluating the touchscreen and digital compass of a mobile device as input methods, and validating their use in different scenarios. Nevertheless, interaction with the DETI Interact system must also be tested. Since many users are not familiar with modern smartphones, and given that touchscreen computers are more common in the scenario explored by this system, DETI Interact was tested on a computer with a touchscreen, as well as the large screens in the department's atrium. The touchscreen computer was not used during the Fitts' Law tests for two reasons: it had a much smaller resolution, which would affect the index of difficulty for each

target, and it allowed users to immediately tap the circle, which would evaluate his/her response time instead of the device's performance.

Interaction with the large screens was done with the smartphone, with the gestures discussed in chapter 4. Its touchscreen is used to control the content presented in each page, while the digital compass is used to alternate between the available pages.

In order to test the system, a list of tasks was prepared for the users to carry out. These tasks consisted of operations of varying difficulty, and ensured that the users would use all the provided features, simulating realistic usage of the system. Users were asked to complete these tasks and rate them regarding their difficulty. Once the tests were over, the users were asked to fill a small questionnaire where they could rate their experience and provide some comments about the system. The proposed tasks were the following:

1. Connect to the system;
2. Find a phone extension for a certain teacher;
3. Find more detailed data for another teacher by accessing his or her website;
4. Check the classroom for a given course;
5. Navigate to a location on Google Earth;
6. Manipulate a three-dimensional model.

The tasks were the same on both computers; however, the Google Earth plugin was not operating correctly on the touchscreen computer, which resulted in the termination of the application. This led to the creation of two different versions of the application, a fully featured version, running on the large screen, and a version without the Google Earth plugin, running on the touchscreen computer. Users on the touchscreen were asked to skip the task referring to navigation using Google Earth.

5.1.2.1 Results

This test was performed with 22 computers engineering students. All the tasks were completed successfully, with both devices showing favourable results. Figure 5.3 shows the average results for the difficulty of each task, with one being very hard, and five being very easy. Apart from the tasks

that the touchscreen could not run – connecting via the mobile application and using Google Earth – it can be seen that the scores between both computers are very similar.

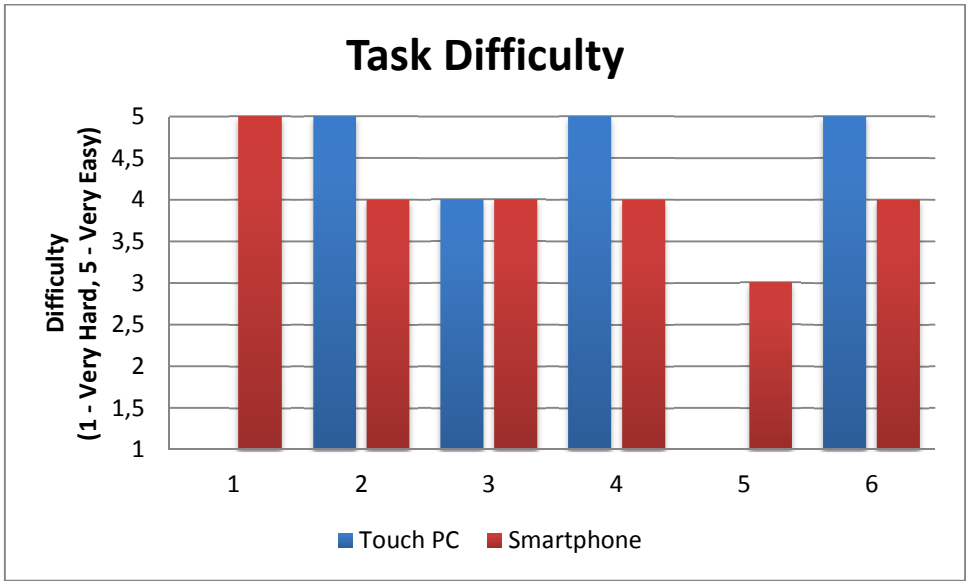

Figure 5.3 – Difficulty of the tasks, expressed by the users, comparing the two interaction methods.

When users connect to the system, a help page is automatically displayed and, in addition, assistance on how to interact with each page is provided at the bottom of the pages. These measures ensure that the users do not feel lost or confused when operating the system. During the observation of the tests, the appropriate gestures were performed. One of the main complications with the mobile device was related with the direction of movement. Since not many users were familiar with modern interaction methods, such as scrolling a list of items on a mobile device, users sometimes expected the system to respond differently. For instance, scrolling down the list of teachers is done by moving the finger upwards on the touchscreen. This simulates the action of dragging the list upwards, in order to reveal the content below. This type of gestures is currently used on most touch-based mobile devices, which led to them being implemented this way on the system. Additionally, users also made mistakes when alternating between pages, which is done by tilting the device sideways. They would either tilt the device in the wrong direction, or attempt to change pages with a horizontal swipe on the touchscreen. Another complication common with most users was interacting with Google Earth. As explained in section 4.2, the pinch gesture was unavailable in the used version of the API, which led to this feature being implemented from scratch. The resulting gesture was very sensitive which left users sometimes feeling out of control.

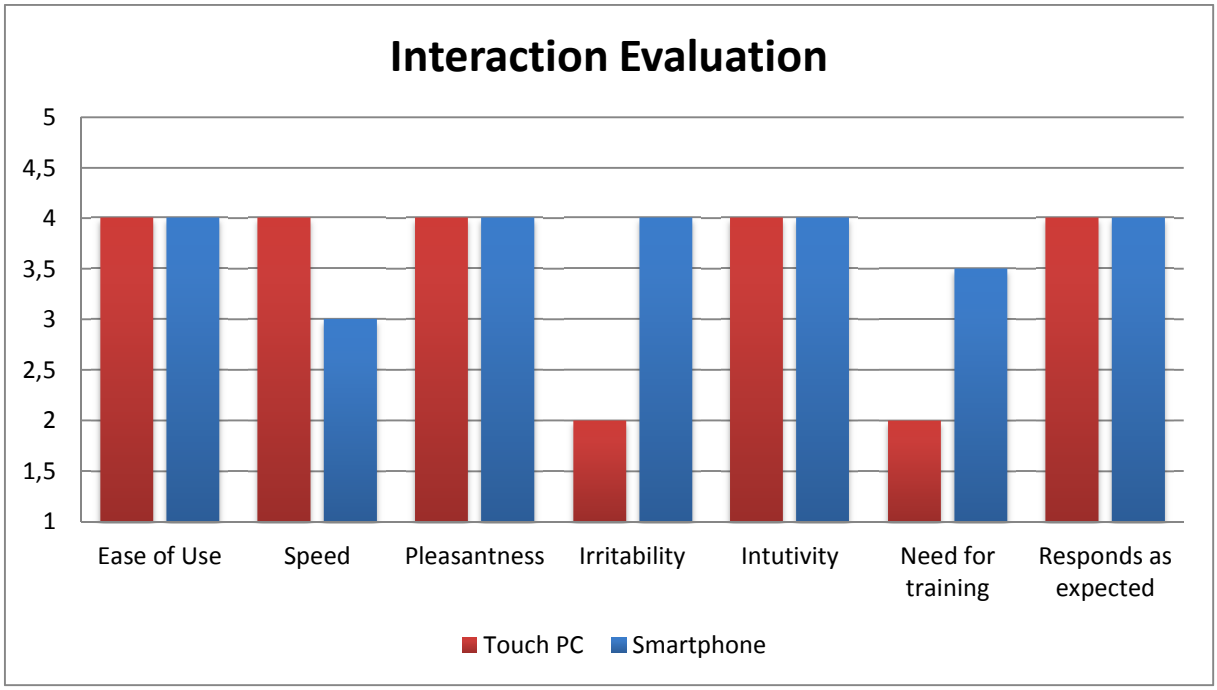

Figure 5.4 - Results of the queries answered by the users after the completion of the tasks.

Following the testing session, users were asked to fill a small survey regarding their experience when using the system. In this survey, they would rate the system concerning ease of use, speed, pleasantness, irritability, intuitiveness, need for training, and responsiveness. The results indicate that the overall experience using the smartphone was inferior when compared to the touchscreen (Figure 5.4). While most users felt that the system was intuitive and easy to use, the lowest scores lie in the speed, irritability and need for training. This, however, was not unexpected. The overall speed of the system was affected by the amount of data being drawn to the screen. As explained on section 3.4.2, the computer running the software presented low performance. With the list of teachers consuming a lot of memory, and the 3D Model Viewer's use of the GPU, which was aggravated by the use of reflection in the code, the system suffered severe hits on performance. In addition, as explained above, some users tried tilting the device, or scrolling in the wrong direction, and most of the users where overwhelmed by the sensitivity of the pinch gesture in the Google Maps page.

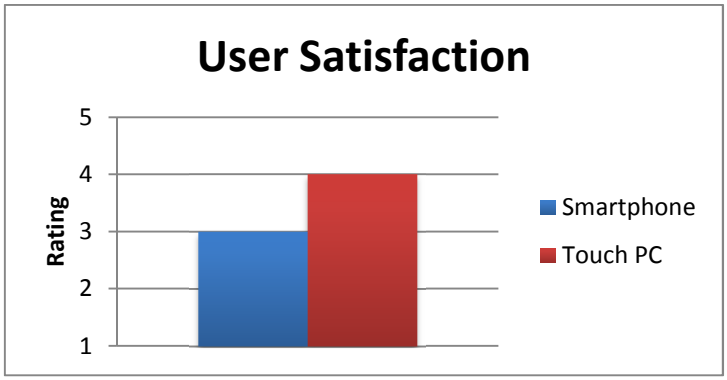

Figure 5.5 – Satisfaction of the users when using either interaction method.

Users were also asked to rate their overall satisfaction with the system (Figure 5.5). Once again, the use of the smartphone received worse ratings. When asked for their preference, users also selected the touchscreen-equipped computer as their favourite, although some users did choose the smartphone over the touch PC (Figure 5.6).

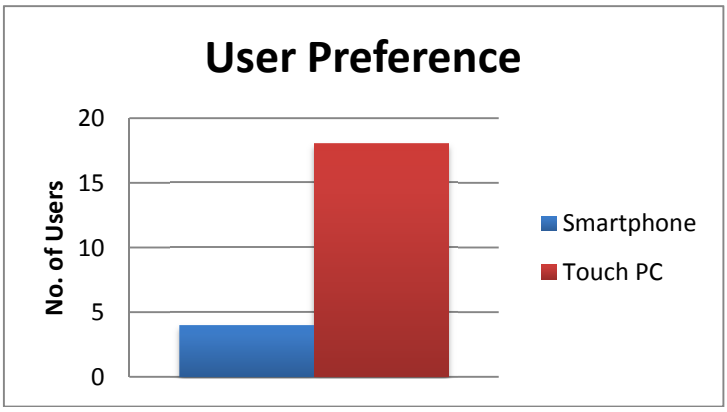

Figure 5.6 – User preference on the interaction method.

In the survey, users had the opportunity to write some comments and give suggestions on how to improve the system. Almost all the comments criticized the sensitivity of the zoom gesture on Google Maps. Some suggested that a double tap gesture could be added to the system to complement the zooming gesture. There were also some references to the titles of the pages, displayed vertically on the sides of the pages, had a very small font. When asked to suggest new features, nearly all the participants requested that the menus for the canteens be displayed on the application. In addition, some news relative to the department, as well as the university, could also be displayed.

5.2 Discussion

Testing the system with users is an important stage when developing software. The objective is to identify errors in the system and determine areas where improvements can be made. For this project, two different testing waves were organized. The first one addressed Fitts' law and its implications. The second involved testing the developed system. Testing was always done with different interaction methods, in order to compare the system with other real world scenarios.

As explained previously, Fitts' Law establishes a relation between the size of a target and its distance from the cursor with the time it takes a user to select it. Data gathered from this test can be used to design better user interfaces. Since DETI Interact proposes the use of a smartphone to control information displayed on a large screen, this test compared the use of the touchscreen and the digital compass with the use of the computer mouse. Both of the studied methods are naturally more arduous when compared with the mouse, but between them, the use of the digital compass offers the worst control method. From the results, it can be concluded that the digital compass should not be used for tasks that require some dexterity. The touchscreen, on the other hand, fares well in these situations.

Considering these results, the interaction methods chosen for DETI Interact – using the touchscreen for most of the operations, and the digital compass to alternate between pages – where considered valid, although some adjustments or alternative gestures could also be provided. In addition, the results also indicate that in scenarios where the user might need to control a cursor or select certain objects, these must have a practical size, with at least 100 pixels in diameter. Furthermore, alternative methods of interaction might be considered. These methods might include different speed or acceleration for the cursor movement, depending on the distance to the target, or using an approach where the targets generate gravity, which would assist the user in the selection.

After processing all the data from the first wave of tests, users were asked to participate in a second wave, in order to evaluate the usability of DETI Interact. While this second wave of tests did not provide the most favourable results towards the system, these were close to those obtained from similar devices (the touchscreen-equipped computer). Considering the results gathered from the previous testing session, concerning Fitts' Law, and the observed reactions of the users, the chosen library of gestures was deemed fitting. While the learning curve for the system may be steeper when compared with using direct interaction methods, users still felt that the system was easy to use, albeit with some initial guidance.

Some users complained about the slow performance of the system. As explained above, this had to do with the implemented 3D Model Viewer, alongside the low performance of the system. This feature was used as a means of comparing three-dimensional interaction techniques using a mobile device equipped with a digital compass, accelerometer, or gyroscope, with direct methods of interaction, such as the touchscreen computer. Given its utility for the department, and the limited availability of 3D models, this feature was removed from the version of the system currently running in the computers of the department. The amount of data displayed on the teachers' page also hinders system performance, although the hit is less severe. Another complaint was concerning the sensitivity of the pinch to zoom gesture on the mobile device. The ideal solution for this problem would be to use the API for Android 2.2, which includes the appropriate functions.

Since the testing, some changes were made to the application, mostly to the user interface such as, some font sizes were increased to improve readability, and the titles of the pages were added to the background of the pages to better situate the users. Visual help was also improved.

6 Discussion

6.1 Conclusion

The main purpose behind DETI Interact was to develop a system that displayed information concerning the Department of Electronics, Telecommunications, and Informatics of the University of Aveiro, and allowed users to interact with this information via an Android smartphone. This would allow users to interact with a system that did not support any type of direct or indirect interaction, such as touchscreens. The system consisted of an application that ran on a computer connected to a large display, which is located in the lobby of the department. An application for Android based devices was also developed which was responsible for the communication between the mobile device and the computer.

6.1.1 Development

As described in chapters 3 and 4, the system communicates with the DSD platform, the department's server that holds information regarding lecturers, students, classes, etc., to gather data relating to the department. It also uses the Google Earth API to display a map plugin and the XNA framework to display and allow interaction with three-dimensional objects.

The desktop application was developed using .NET and WPF. These technologies were helpful in prototyping and developing the system rapidly, as well as offering a greater performance. Given the low hardware specifications of the target machines, it might have been worthy to attempt to develop the system in a lower level language (using C++ with QT for the user interface, for example). An

attempt was made using Windows Forms, but the large amount of data drawn on screen was significantly worse, when compared to WPF. Using WPF was also helpful in developing a system that was easily modifiable and extensible, for its use of the MVVM pattern.

The Android mobile application was developed in Java. While Android offers a good development environment when compared to other mobile platforms, its programming model, using the MVC Pattern, and the provided API, feels cluttered, which degraded the developing experience. The main problems were encountered with the Bluetooth communication. It was chosen for its simplicity and lack of configuration needed by the user, but the API provided in the Android SDK damaged the experience that could be provided to the users of DETI Interact.

Overall, the development of DETI Interact did not face any major setbacks, due to well-defined system requirements and suitable choice on the development platforms.

6.1.2 Results

The system was deployed to the lobby where it has been running since January 2011. During this time, it has been receiving updates and generating usage logs, which assisted in its development and maintenance.

The system was later tested with users to evaluate the performance of the smartphone as an input peripheral, as well as the usability of the application. The first testing wave targeted the evaluation of the performance of the smartphone. It was based upon Fitts' Law and yielded interesting results, which are described in detail in Chapter 5. The results suggested that the use of the touchscreen of a mobile device could be an appropriate input peripheral, while the use of the digital compass could be more problematic. This corroborated the choices made during development, where the compass was used for switching pages, by tilting the device sideways, and the touchscreen was used to interact with the content presented on each page.

A second wave of testing evaluated the usability of DETI Interact. Users were given a series of tasks to fulfil and then rate their difficulty. In the end, they also participated on a small survey where they rated their experience and commented the system. The results were compared with the use of a touchscreen-equipped computer, which would be a more familiar device. While the touchscreen computer received better results, these remained close to the results obtained from the smartphone. Users thought that, while the system was easy to use, some usage scenarios were

more irritating in the smartphone, and that the learning curve was steeper. While at first glance the results seem unfavourable towards the smartphone, it must be taken into account that these were compared with other already well-established methods of interaction, such as the computer mouse, with which users are more experienced. However, the mouse itself took thirty years to become a ubiquitous device (Buxton 2007). Natural User Interfaces (NUI), such as those explored by voice, gesture, and touch, albeit quickly becoming embraced by the industry and the public, are still struggling to become ubiquitous. They are still criticized for not being as good as a computer mouse for most tasks, even though they might be superior in other tasks. As William Buxton says: "everything is best for something and worst for something else", also adding, "the trick is knowing what is what, for what, when, for whom, where, and most importantly, why. Those who try the replace the mouse play a fool's game. The mouse is great for many things. Just not everything" (Buxton 2007). It was not unexpected that the interaction performance with the mobile device would be worse, but it allows for a good method of interaction in a situation where other alternatives would not be viable or would require a greater investment.

In conclusion, DETI Interact proposed to study approaches to interaction with large displays, and implement and information system on the lobby of the Department of Electronics, Telecommunications, and Informatics of the University of Aveiro. Although the results could have been superior, the system received good reviews, and is currently running on a on a computer in the department. While direct manipulation of data on a remote screen via a smartphone might appear to be an arduous task for users, given the impossibility of using a computer mouse, an appropriate user interface and set of gestures can make a system usable, with a promising user experience.

6.2 Future Work

As with any software project, there are always areas that can be improved, and features that can be added. The results and comments gathered from the testing sessions were invaluable in finding areas in need of improvement. The principal complaint about the system was in regards to the zoom gesture in the Android device. This had to do with the gesture not being available in the API and having to be written from scratch. While the sensibility can be improved, version 2.2 of the API includes the classes that handle this gesture, which would provide the best user experience.

In addition, alternative gestures can be implemented, such as the Double Tap, also a common in zooming tasks. Furthermore, the system could use and extended gesture library. For instance, some

users would try to use the Fling gesture to alternate between pages, instead of tilting the device. Since the touchscreen allows detection of multiple points, a two-finger swipe can be implemented.

In respect to new features, users suggested that the system could also display the menus for the canteens, ticket information from the university's administration services, as well as news from the department. Additional information such as exam dates, vacant classrooms, integration with the university's library, was also suggested by other students.

Apart from the functionalities suggested by the users, other features were already being considered. One interesting feature would be to make the system context-aware. Once a user connected to the system, he would be identified and specific information could be displayed, such as his timetable, exam dates, project deliveries, etc. This would require further integration with the university's systems such as PACO (*Portal Académico Online*) and Moodle. Additional information could also be displayed on the mobile device's screen. The mobile device is currently used solely as an input peripheral, but the available screen can also be used for information display. As stated in chapter 4, Android 2.3 allows for the use of Insecure Bluetooth Sockets. This removes the need for the devices to pair, which would improve the user experience on the first connection to the system. However, using a more recent version of Android (whether it is 2.2, for the scale gesture, or 2.3, for the insecure Bluetooth sockets) will exclude users running older versions.

In addition, other mobile platforms could also be included. While not all provide access to the Bluetooth stack, future updates could bring such functionality, and other communication methods are available (e.g. WiFi, Sockets). Furthermore, given the results obtained from the first testing session, other interaction methods could be added to the system, such as the ability to control the mouse cursor. This would allow users to navigate webpages, for example, as opposed to simply viewing them as is currently done. It could also improve selection-based tasks, such as those existent in the teacher page. DETI interact was developed in a way that interaction with the system is not dependent on any device, which allows for the system to support additional devices such as video cameras or the Microsoft Kinect device. The system could also be extended in order to allow multiple users in a collaborative environment, as well as the possibility of spanning across additional displays. Augmented reality is also an option that can enrich the experience of the user. Testing also revealed that the mobile device offers a suitable method of interaction for 3D content, meaning that further applications of three-dimensional content should be studied.

References

Anthony, S. (2010). "Microsoft Kinect controlling Windows 7, exciting proof of concept." Retrieved 25/11/2010, from http://downloadsquad.switched.com/2010/11/22/microsoft-kinect-controlling-windows-7-exciting-proof-of-concep/.

Boring, S., D. Baur, et al. (2009). "Touch Projector: Mobile Interaction through video." Proceedings of CHI 2010, Atlanta, GA, April 10-15: 2287-2296.

Buxton, W. (2007, 21/03/2011). "Multi-Touch Systems that I Have Known and Loved." Retrieved 18/06/2011, from http://www.billbuxton.com/multitouchOverview.html.

Campos, D. (2008). Distribuição de Serviço Docente: back e front office Web. Departamento de Electrónica, Telecomunicações e Informática, Universidade de Aveiro. **MSc:** 148.

CellPhonesMarket. (2011). "Google's new SDK features an API for "insecure Bluetooth socket connections"." Retrieved 21/05/2011, from http://www.cellphonesmarket.com/news/googles-sdk-features-api-insecure-bluetooth-socket-connections/.

Cheng, K. and K. Pulo (2003). Direct interaction with large-scale display systems using infrared laser tracking devices. Proceedings of the Asia-Pacific symposium on Information visualisation - Volume 24. Adelaide, Australia, Australian Computer Society, Inc.**:** 67-74.

Cull, T. (2010). "Researchers Highlight Recent Uptick in Java Security Exploits." Retrieved 14/02/2011, from http://www.infoq.com/news/2010/10/java-exploit-uptick.

DailyTech. (2011). "Android Overtakes Symbian as World's No. 1 Smartphone Platform." Retrieved 14/02/2011, from http://www.dailytech.com/Android+Overtakes+Symbian+as+Worlds+No+1+Smartphone+Platform/article20787.htm.

Finke, M., A. Tang, et al. (2008). Lessons learned: game design for large public displays. Proceedings of the 3rd international conference on Digital Interactive Media in Entertainment and Arts. Athens, Greece, ACM**:** 26-33.

Fling, B. (2009). Mobile Design and Development, O'Reilly.

Foot, P. "In The Hand .NET components for mobility." Retrieved 04/11/2010, from http://inthehand.com/.

GIZMODO. "Orange Shows Off Gesture Based Interaction Screen, Touch Screens Look On in Horror." Retrieved 04/11/2010, from http://gizmodo.com/346845/orange-shows-off-gesture-based-interaction-screen-touch-screens-look-on-in-horror.

Google. (2005). "Google Earth." Retrieved 13/06/2011, from http://www.google.com/earth/index.html.

Google. (2010). "Features - Google TV." Retrieved 24/11/2010, from http://www.google.com/tv/.

Google. (2011). "What is Android? | Android Developers." Retrieved 12/04/2011, from http://developer.android.com/guide/basics/what-is-android.html.

Hardy, R. and E. Rukzio (2008). Touch \& interact: touch-based interaction of mobile phones with displays. Proceedings of the 10th international conference on Human computer interaction with mobile devices and services. Amsterdam, The Netherlands, ACM**:** 245-254.

Hardy, R. and E. Rukzio (2008). Touch \& Interact: touch-based interaction with a tourist application. Proceedings of the 10th international conference on Human computer interaction with mobile devices and services. Amsterdam, The Netherlands, ACM**:** 531-534.

Ideum. (2011). "Supported Multitouch Gestures." Retrieved 16/06/2011, from http://gestureworks.com/support/supported-gestures/.

Kaviani, N., M. Finke, et al. (2009). "Encouraging Crowd Interaction with Large Displays using Handheld Devices " Proceedings of CHI 2009, Boston, MA, April 4-19.

MacKenzie, I. S. and W. Buxton (1992). Extending Fitts' law to two-dimensional tasks. Proceedings of the SIGCHI conference on Human factors in computing systems. Monterey, California, United States, ACM**:** 219-226.

Madhavapeddy, A., D. Scott, et al. (2004). "Using Camera-Phones to Enhance Human-Computer Interaction." Proceedings of Ubiquitous Computing (Adjunct Proceedings: Demos): 1-2.

Microsoft. (2009). "Microsoft .NET Framework." Retrieved 13/06/2011, from http://www.microsoft.com/net/.

Microsoft. (2010). "Introduction to WPF." Retrieved 20/08/2010, from http://msdn.microsoft.com/en-us/library/aa970268.aspx.

Microsoft. (2010). "Kinect - Xbox.com." Retrieved 25/11/2010, from http://www.xbox.com/en-US/kinect.

Microsoft. (2011). "Kinect for Windows SDK from Microsoft Research." Retrieved 16/06/2011, from http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/.

Microsoft. (2011). "Microsoft Expression Blend® 4 | Silverlight | Rich Internet Applications | XAML | WPF Applications | .NET Platform | TFS | VB | C# | Microsoft® Expression®." Retrieved 13/04/2011, from http://www.microsoft.com/expression/products/Blend_WhatIsExpressionBlend.aspx.

Microsoft. (2011). "Silverlight 5 beta: The Official Microsoft Silverlight Site." Retrieved 13/06/2011, from http://www.silverlight.net/getstarted/silverlight-5-beta/.

Microsoft. (2011). "Touch." Retrieved 25/05/2011, from http://msdn.microsoft.com/en-us/library/cc872774.aspx.

Mihailescu, D. (2010). "Benchmark start-up and system performance for .Net, Mono, Java, C++ and their respective UI." Retrieved 14/02/2011, from http://www.codeproject.com/KB/dotnet/RuntimePerformance.aspx.

Miyaoku, K., S. Higashino, et al. (2004). C-blink: a hue-difference-based light signal marker for large screen interaction via any mobile terminal. Proceedings of the 17th annual ACM symposium on User interface software and technology. Santa Fe, NM, USA, ACM**:** 147-156.

Morril, J. (2008). "XNA, Meet WPF." Retrieved 04/11/2010, from http://jmorrill.hjtcentral.com/Home/tabid/428/EntryId/259/XNA-Meet-WPF.aspx.

Palha, R. (2010). Interacção entre um dispositivo móvel e um ecrã de grandes dimensões. Departamento de Electrónica, Telecomunicações e Informática, Universidade de Aveiro. **MSc**.

PhoneGap. (2008). "PhoneGap." Retrieved 11/04/2011, from http://www.phonegap.com/.

Richter, J. (2010). CLR via C#, Microsoft Press.

Samsung. (2010). "Samsung Internet @ TV." Retrieved 24/11/2010, from http://www.samsung.com/uk/internet-tv/.

Seewoonauth, K., E. Rukzio, et al. (2009). Touch & connect and touch & select: interacting with a computer by touching it with a mobile phone. Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services. Bonn, Germany, ACM**:** 1-9.

Stogaitis, M. and M. Sun. (2008). "Gmote - Android Remote." Retrieved 21/11/2010, from http://www.gmote.org/.

Tang, A., M. Finke, et al. (2008). Designing for bystanders: reflections on building a public digital forum. Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems. Florence, Italy, ACM**:** 879-882.

TheAlternative. "The Alternative Homepage." Retrieved 21/11/2010, from http://www.thealternative.co.uk/.

Thurrot, P. (2010). "More work on Chapter 4 and notes about the origins of Metro." Retrieved 14/05/2011, from http://windowsphonesecrets.com/2010/04/04/more-work-on-chapter-4-and-notes-about-the-origins-of-metro/.

Thurrott, P. (2010). "Xbox 360 Dashboard Screenshots Gallery 1." Retrieved 14/05/2011, from http://www.winsupersite.com/article/product-review/xbox-360-dashboard-screenshot-gallery-1.

Totilo, S. (2010). "Natal Regonizes 31 Body Parts, Uses Tenth of Xbox 360 "Computing Resources"." Retrieved 01/03/2011, from http://kotaku.com/5442775/natal-recognizes-31-body-parts-uses-tenth-of-xbox-360-computing-resources.

Underkoffler, J. (2010). "John Underkoffler points to the future of UI." Retrieved 01/03/11, from http://www.ted.com/talks/john_underkoffler_drive_3d_data_with_a_gesture.html.

Vogel, D. and R. Balakrishnan (2004). Interactive public ambient displays: transitioning from implicit to explicit, public to personal, interaction with multiple users. Proceedings of the 17th annual ACM symposium on User interface software and technology. Santa Fe, NM, USA, ACM**:** 137-146.

W3C. (2004). "Web Service Glossary." Retrieved 04/11/2010, from http://www.w3.org/TR/ws-gloss/.

Wired. (2009). "5 Nifty Apps That Turn Your Android Into A Universal Remote." Retrieved 21/05/2011, from http://www.wired.com/gadgetlab/2009/11/5-nifty-remote-apps/.

Appendix I – User Tasks and Questionnaire

After the tests conducted on DETI Interact, users were asked to fill the following questionnaire:

DETI Interact: Questionário do Utilizador

Instruções: Agradecemos a sua colaboração na realização deste estudo, que tem por objectivo avaliar a *Interface de Utilizador* do *DETI Interact* e, consequentemente, tentar melhorá-la seguindo os critérios de *Usabilidade*.
A sua colaboração constitui um factor importante para o êxito desta avaliação, por isso solicitamos-lhe o preenchimento deste questionário, cujos dados serão usados com total anonimato apenas para fins científicos.

1. Dados pessoais

Nº Mecanográfico: _____ Turma:_____ Data:_____/_____/_____
Género (M/F): _____ Idade: _____ Nº de Utilizador: _____

Tem experiência com dispositivos que possuem ecrã sensível ao toque, acelerómetro ou bússola digital (por exemplo *smartphones*)? S___ N___

Se respondeu Sim:

Que tipo de aplicações usa com as tecnologias referidas?
Navegação no sistema___ Escrita de mensagens___ Jogos___

Outras:_____

Usa: Várias horas por dia___ várias horas por semana___ várias horas por mês___
Várias horas por ano___ outro tipo de utilização:_____

2. Opinião geral sobre o *DETI Interact*

Após a utilização do sistema e tendo em conta a sua avaliação final, preencha o círculo que melhor reflecte a sua opinião em relação à utilização da ferramenta. Caso considere que estas quantificações não são aplicáveis, escolha NA.

2.1. Opinião sobre a utilização do *DETI Interact* *(preencha o círculo da opção que melhor corresponde à sua posição)*

2.1.1 Interacção através do **Smartphone:**

É fácil orientar-me no *DETI Interact*.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A velocidade de navegação é adequada.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA

A navegação é agradável.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A navegação tem algumas características irritantes.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A utilização do *DETI Interact* com esta forma de interacção é intuitiva.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A utilização do *DETI Interact* com esta forma de interacção requer treino.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A utilização do *DETI Interact* com esta forma de interacção é intuitiva.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA

2.1.2 Interacção através do Computador **Touch:**

É fácil orientar-me no *DETI Interact*.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A velocidade de navegação é adequada.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A navegação é agradável.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A navegação tem algumas características irritantes.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A utilização do *DETI Interact* com esta forma de interacção é intuitiva.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A utilização do *DETI Interact* com esta forma de interacção requer treino.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA
A utilização do *DETI Interact* com esta forma de interacção é fácil.	Discordo totalmente	○	○	○	○	○	Concordo totalmente	NA

3. Por favor indique o grau de satisfação na utilização de cada uma das formas de interacção numa escala de 1 (muito baixo) a 5 (muito alto):

Smartphone:	1 nada satisteito	○	○	○	○	○	5 muito satisfeito

Computador Touch:	nada satisteito	○	○	○	○	○	muito satisfeito

4. Por favor assinale com qual das formas de interacção preferiu navegar no *DETI Interact*:

Smartphone: ☐ ☐ ***Touch***

5. Se pretender pode deixar aqui comentários sobre as formas de interacção usadas no *DETI Interact*:

_____
_____
_____
_____
_____

6. Que outros serviços gostaria de ver integrados no DETI Interact.

_____
_____
_____
_____
_____

FIM

Muito obrigada pela sua colaboração!