Maciej Gryka

[ee51mg@surrey.ac.uk](mailto:ee51mg@surrey.ac.uk)

URN: 1553283

3D Visualisation of Sensor Information on Google Android Platform

Final Year Project

Interim Report

The following is a midterm report presenting an initial phase of the “3D Visualisation of Sensor Information on Google Android Platform” final year project work. It contains information about the project itself, its objectives and requirements as well as literature review and background information on tools and techniques used. Furthermore information is given on the tasks completed so far and a plan for further activities necessary to complete the project.

Contents

[1. Introduction 1](#_Toc220299964)

[1.2 Google Android 1](#_Toc220299965)

[1.3 Tools 2](#_Toc220299966)

[1.3.1 Android SDK 2](#_Toc220299967)

[1.3.2 Eclipse IDE 2](#_Toc220299968)

[2. Project Description 2](#_Toc220299969)

[2.1 Project Objectives 2](#_Toc220299970)

[2.2 Expected Outcomes and Testing 3](#_Toc220299971)

[3. Literature Review 4](#_Toc220299972)

[3.1 Methods of Gathering Information 4](#_Toc220299973)

[3.2 Choosing Software Development Methodology 4](#_Toc220299974)

[3.2.1 Pure Waterfall 5](#_Toc220299975)

[3.2.2 Modified Waterfalls 5](#_Toc220299976)

[3.2.3 Design-to-Schedule 6](#_Toc220299977)

[3.3 Similar Projects 6](#_Toc220299978)

[3.3.1 API Demos 7](#_Toc220299979)

[3.3.2 AndroidGL 7](#_Toc220299980)

[3.3.3 Suhas3D 7](#_Toc220299981)

[3.4 Android Technical Information 8](#_Toc220299982)

[3.5 Background Information on OWL 9](#_Toc220299983)

[4. Requirements 10](#_Toc220299984)

[5. Work Record 13](#_Toc220299985)

[Week 1: 13](#_Toc220299986)

[Week 2 13](#_Toc220299987)

[Week 3 14](#_Toc220299988)

[Week 4 16](#_Toc220299989)

[Week 5 16](#_Toc220299990)

[Week 6 16](#_Toc220299991)

[Week 7 17](#_Toc220299992)

[Week 8 17](#_Toc220299993)

[Week 9 17](#_Toc220299994)

[6. Future Work 19](#_Toc220299995)

[6.1 Development 19](#_Toc220299996)

[6.2 Final Report 20](#_Toc220299997)

[7. Conclusions 20](#_Toc220299998)

[References 22](#_Toc220299999)

[Appendix 1 24](#_Toc220300000)

# 1. Introduction

"3D Visualisation of Sensor Information on Google Android Platform" is a final year project undertaken by a student on a BEng Electronics & Computer Engineering course at University of Surrey in Guildford, UK. It aims to design, develop and document an application that will run on Google Android mobile operating system.

The aim of the application, on the highest level, is to communicate with a data repository that stores the information about sensor readings in the BA building on the University of Surrey's campus and represent the obtained data on a 3D model on the device's screen, thus visualising the current state of affairs in the building's rooms, such as lighting, temperature, occupancy etc.

To gain a complete appreciation of the project it is necessary to be familiar with some background information, such as what Google Android is, knowledge of object oriented programming principles and tools used to build the application (Android SDK, Eclipse IDE and others) - these are explained in this chapter and this knowledge is built upon further in the report.

## 1.2 Google Android

Google Android is described on its official website as follows:

*"The Android Platform is a software stack for mobile devices including an operating system, middleware and key applications. Developers can create applications for the platform using the Android SDK. Applications are written using the Java programming language and run on Dalvik, a custom virtual machine designed for embedded use, which runs on top of a Linux kernel."* (Google, 2008).

It is one of the projects of Open Handset Alliance (<http://www.openhandsetalliance.com/>) that aims to produce an open handset standard that includes both hardware and software solutions. Android is the software part of it and will enable developers and end-users to build and use the same operating system and applications on different devices from variety of manufacturers.

Although this application will not be suitable to be released to the wider audience, it is worth noting that the Android Market, part of the Android project, is a great way to distribute Android-compatible software. It enables developers to release developed software at almost no cost and is also an effective marketing channel due to the large audience.

Android is a one-of-a-kind solution with which the OHA aims to become a significant player in the smartphone market. Its openness is the biggest differentiator from the other products in the field, and it enables much greater developer community and thus innovation.

## 1.3 Tools

### 1.3.1 Android SDK

Google provides Android Software Development Kit to aid developers in producing Android applications. It is consists of APIs (Application Programming Interfaces) that enable easy interaction with device's hardware (buttons, touch screen, compass, GPS etc.) and operating system. It also includes a plug-in for Eclipse IDE that enables easy debugging and simulation of the written applications. At the time of writing the report, the latest version was the Android 1.0 SDK release 2.

### 1.3.2 Eclipse IDE

Eclipse IDE is *"an open source, robust, full-featured, commercial-quality, industry platform for the development of highly integrated tools and rich client applications"* (Eclipse Foundation, 2008).

It is widely used in industry and highly customisable, especially with the use of plug-ins (such as the one developed for Android).

# 2. Project Description

## 2.1 Project Objectives

The aim of the project is to design and develop an application for Google Android operating system. The application should be able to interpret data stored in a repository and represent it graphically in the three-dimensional environment. Interpreted data will contain information about sensor readings from within a single building and will be stored in an OWL (Web Ontology Language) format.

Specific objectives are as follows:

* Application will be written using Android SDK v1 (or higher, if available) in Java and should run on all Android-powered devices.
* The application will show a 3D representation of provided data mapped onto a model of a building. The data will contain information about sensor readings in different rooms in the building. Each set of readings for any given room will be represented as a set of icons on the room's wall and other indicators (such as lighting).
* Data will be automatically refreshed whenever database is changed (automatic notification from the database will be sent to the device to refresh data). It will also be possible to refresh the data upon user request. Refresh process should take less than 10 seconds.
* User should be able to freely navigate in the 3D space. That means ability to view object from different perspectives (rotate around X and Y axes) and to zoom in/out on the specific point.
* Optionally, there should be a possibility for the user to set values for given actuators and write them to the repository. This feature will be added if time and resources permit.
* Input (and optionally output) will be in an OWL format. Data in the repository that the application will connect with will be stored in OWL.
* Design and documentation for this project are crucial and are also success indicators. Design should be sufficiently documented and carried out in a well-structured way, following best software development practices.

## 2.2 Expected Outcomes and Testing

By the end of the project a working application is expected to be developed. All requirements should be fulfilled. The application should be fully functional and usable with no major issues or bugs.

If time permits, some of the optional requirements may be fulfilled as well, such as the possibility to update the repository from the device.

The tests are based on the requirements, so that each test will determine whether a specific requirement, or a set of them, is met or not:

* + Func1: Ability to show a 3D representation of data
    - Check if the application shows the model of the building with appropriate indicators (icons) in appropriate places.
  + Func2: Ability to pull and present data from the database in real time.
    - Change an entry in the data repository and monitor the device whether it updates accordingly and how long it takes.
    - Try to refresh the data manually (option in the menu).
  + UI Requirements
    - Check whether UI is intuitive and behaves in the way specified in the requirements
  + Navigation
    - Try to navigate around in the 3D space and determine whether it is easy to look at the model from different perspectives and look into each room

# 3. Literature Review

## 3.1 Methods of Gathering Information

Because Android is a relatively new product there are not many books available on the subject and the ones released are very general. There is, however, a vast amount of detailed information on the Internet - official, Google-run websites as well as private, community-run ones. Therefore, while information on general topics, such as project management, software development methodologies, object-oriented programming principles and best practices are taken from books and journals, Android-specific and project-specific information is mainly obtained from the Internet.

## 3.2 Choosing Software Development Methodology

During any software development project it is essential to collect the requirements before designing the software, to have a design before implementation etc. – namely to follow a structured software development lifecycle. As circumstances vary, different lifecycles have been developed and described over the years to accommodate different needs of different projects. In this section, several main methodologies are presented and are followed by a short discussion on why they are suitable for this project.

Below is only a summary of some common methodologies. For more detailed information, the reader could refer to *Rapid Development: Taming Wild Software Schedules* (McConnell, 1996, pp. 133 – 161), where this subject is explained in much greater detail.

### 3.2.1 Pure Waterfall

Waterfall model is the oldest and the most well-known of all described here. It is probably the most intuitive one and that serves as the basis for most other models, even though many problems have been recognised since its introduction by Royce (1987). The basic idea of this model is to divide development into few separate stages (like requirements analysis, design, coding, testing) and go through them in order. Each stage can only be started if the previous one is completed, i.e. coding would not start before the design is finished and approved.

Because of the rigid structure this model is often used in formal projects that carry a high risk, for instance military projects and carries a lot of documentation. In order for pure waterfall model to be suitable, requirements need to be well-understood and stable throughout the whole cycle.



Figure : Pure Waterfall model as found in Royce (1987, pp. 329)

### 3.2.2 Modified Waterfalls

Despite its advantages it is usually better to use a variation of the pure waterfall model to address some of the major flaws. Described below are two changes that address its inflexibility and that will be adopted in this project:

* + “Sashimi” i.e. overlapping phases is a model similar to the pure waterfall, with the exception of starting time of each stage. In sashimi model phases overlap slightly, so the design is started before the requirements are finalised, coding before the design is completed etc. That accommodates situations, where it is difficult to predict various problems, for instance when during coding it becomes apparent that a design decision was wrong. The name “sashimi” comes from the fact that the graph representing this particular model resembles the Japanese dish.

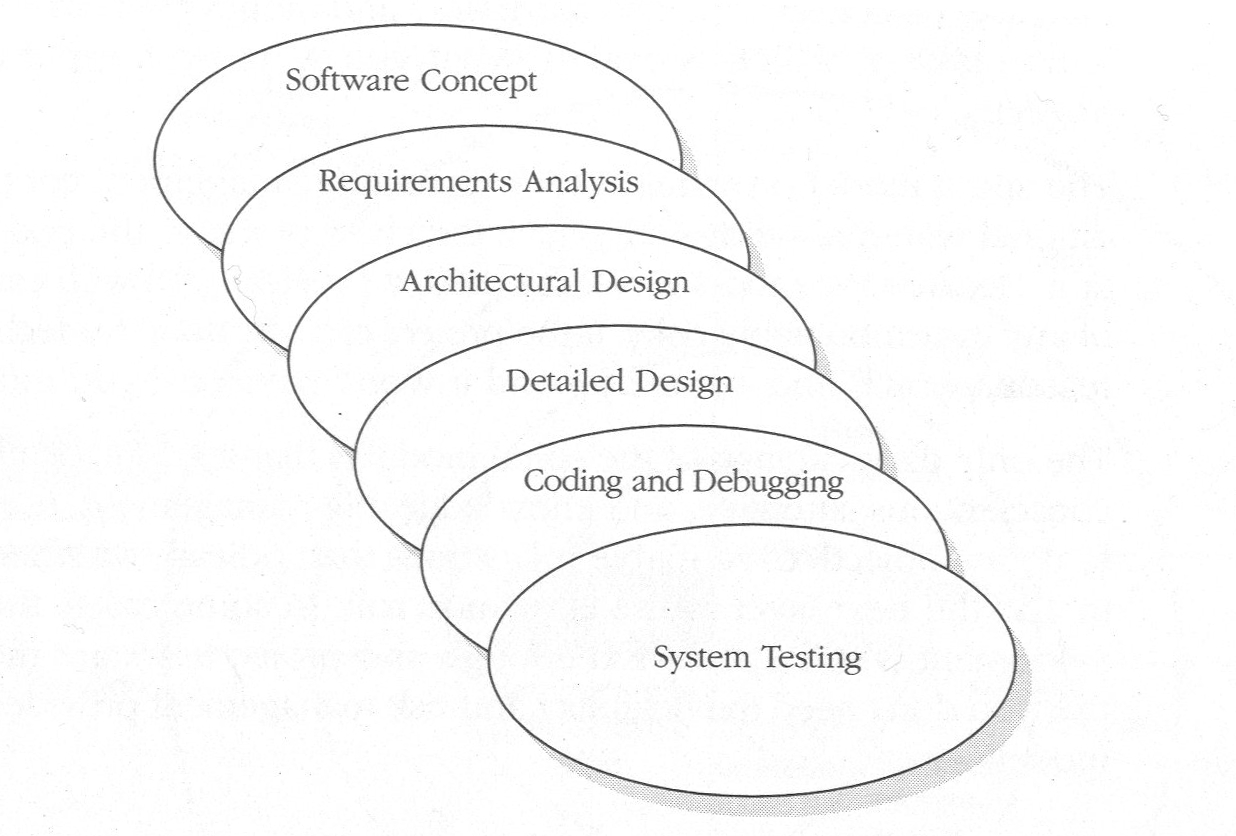


Figure : Sashimi model as found in McConnell (1996, pp. 144)

* + Allowing for more regression (coming back to stages that have already been gone through), which is called “risk reduction” by McConnell (1996), allows to revise the stages that have been already gone through, taking into consideration how some of the later stages progressed.

### 3.2.3 Design-to-Schedule

This model is suitable for projects, where delivering a product by the given deadline is the critical success factor – even if the product doesn’t have all of the specified features. It is based on several releases, each one with lesser priority features than the previous one, so that the most important requirements are satisfied first. Subsequent releases are made until all features are included, or until the time runs out in which case only the low-priority characteristics are missing.

Design-to-Schedule model is best suited for situations, where the deadline is unmovable and it is hard to predict, whether all the work will be completed by then, making it a good choice for this project.

To summarise, this project will use a waterfall model with overlapping stages and risk reduction (iterating over previous stages) as well as planning for few releases, each with lower priority than the previous one.

## 3.3 Similar Projects

The author was unable to find Android projects that would be useful for this topic described in any publication such as a journal or a book. This is probably partly due to the fact that Android is still a young platform and also because it is far easier to distribute such projects electronically, for instance via an Internet website. Therefore, the following projects all come from various referenced websites and while some of them (like Google’s *API Demos*) are very well-documented, others only include the source code within an Eclipse project with no documentation at all. All of them, however, proved useful in illustrating how to achieve some of the tasks needed for this project.

### 3.3.1 API Demos

This Google-developed project acts as a sample of all the basic operations on Android. It covers a wide range of topics, from simply displaying text on the screen to using dialog boxes to managing notifications to 3D graphics.

For this project, *Graphics\OpenGL ES* part of this sample is of particular interest. It illustrates the most basic ways of working with the OpenGL ES API, from creating a scene to drawing static and animated objects. It also includes texturing.

While useful to gain familiarity with Android’s 3D capabilities, however, this code only uses simple models that are built programmatically, like cubes and triangles. Because of complexity of the model in this final year project, this approach in not suitable and a way needs to be found of importing externally-built models and using them.

### 3.3.2 AndroidGL

*AndroidGL* consists of eleven tutorials, which present basic OpenGL ES functionality. While first few of them are comparable to the examples found in the *API Demos*, later ones go into more depth. Concepts such as animation, lighting and reflections are implemented, however, with minimal documentation (few comments in the code as well as Javadoc).

Each of the tutorials, presents a single concept and the author have found them useful in familiarising with OpenGL and its capabilities. However, even though the libraries include an .md2 loader that should be able to load external files into OpenGL ES, no tutorial actually uses it.

### 3.3.3 Suhas3D

The following project does exactly what *API Demos* and *AndroidGL* were missing in terms of functionality. It loads an .md2 file with a 3D model and animates it on the Android platform. Although there is not much documentation, it is still very useful and allows for code reuse.

The downside is that the animation appears to be very slow. It is not certain at this point, whether it is only because of emulator’s limitations (it does not use hardware graphics acceleration) or it is caused by an un-optimised code or device’s capabilities. Testing will need to be performed with the real device to determine this.

## 3.4 Android Technical Information

Android, as mentioned before, is a complete software stack that enables development of applications to control almost every aspect of how the device works. The architecture of the system can be seen on Figure 3.



Figure 3: Android System Architecture as presented by Google (2008) at:   
http://code.google.com/android/what-is-android.html

Starting from the bottom of the stack, different tiers include:

* + Linux Kernel takes care of low-level memory and power management as well as contains drivers to interact with the hardware. It uses Linux version 2.6
  + Android Runtime provides core Java libraries to be used by developers as well as Dalvik virtual machine support. Dalvik is optimised for mobile devices (minimal memory usage) and each application runs on a separate instance (separate virtual machine).
  + Libraries provide support for non-core, but essential capabilities like media framework, graphic libraries, database management and web browser engine. These are C/C++ libraries, as opposed to Java core libraries.
  + Application Framework introduces high-level classes used to build applications. They enable “black-box” approach to device’s features and another abstraction layer. Few components worth mentioning here are:
    - Activity (android.app.Activity) – this is what user interacts with. Activity can be a screen that is presented to the user, or it can also be a member of another activity. It receives user’s actions and acts upon them.
    - Intent (android.content.Intent) is what activity sends to perform and action. It can be seen as a notification to the operating system that an action is to be performed. Intents are also used to start activities.
    - Content Providers allow access to various resources like databases, images etc.
  + Applications are what users interact with to perform some action. All the phone functions for instance (making and receiving calls, text messages etc.) are applications allowing developers to modify or even replace them if necessary. This project, when completed, will be an application in that sense.

## 3.5 Background Information on OWL

OWL stands for Web Ontology Language. To understand what OWL is and why it was invented, first it is necessary to know what ontology means. In philosophical sense (i.e. original meaning of the word) ontology concerns the nature of existence of things and relationships between them. In computer science terms, Web Ontology Language tries to create a machine-readable format of data that describes not only entities and their properties, but also relationships between such entities.

Another, older standard, RDF serves similar purpose and OWL is based on it. There are, however, some significant differences: OWL is a variation of RDF that has more strict rules and is also more machine-interpretable. OWL uses XML file format to represent the data and is a recommendation of W3C (World Wide Web Consortium).

The purpose of OWL is to create “Semantic Web” that will be able to analyse data and draw conclusions that standard tools would be unable to. It aims to create a data model that will allow machine interpretation closer to the one exhibited by humans.

# 4. Requirements

Below is the list of requirements for the application. These were documented by the student after a discussion and an agreement with the customer on the application’s features and behaviour. Below list is a more detailed description of the application that follows from the objectives noted in the second chapter. It also forms the basis for the tests that shall be conducted after development.

|  |  |
| --- | --- |
| **ID** | Func1: Ability to show a 3D representation of data |
| **Area** | Functional |
| **Description** | The application will show a 3D representation of provided data mapped onto a model of a building. The data will contain information about sensor readings in different rooms in the building. Each set of readings for any given room will be represented as a set of icons on the room's wall and other indicators (such as lighting). |

|  |  |
| --- | --- |
| **ID** | Func2: Ability to pull and present data from the database in real time. |
| **Area** | Functional |
| **Description** | Data will be automatically refreshed whenever database is changed (automatic notification from the database will be sent to the device to refresh data). It will also be possible to refresh the data upon user request.  Refresh process should take less than 10 seconds. |

|  |  |
| --- | --- |
| **ID** | UI1: Welcome Screen |
| **Area** | UI |
| **Description** | After the application is initialised user will be presented with a welcome screen. By default, it would be the 3D space centred onto the model of the building. When no data is present, the model would still be presented, but with no sensor indicators. Also a notice/icon should be visible (somewhere in the corner of the screen) saying that no data is present). |
| **Notes** | We assume that building model is static, i.e. built into the application ans so we don't need to get it from anywhere and can display even if not connected. |

|  |  |
| --- | --- |
| **ID** | UI2: Main Menu |
| **Area** | UI |
| **Description** | Following options will be presented in the main menu (after pressing "menu" button on the device):   * Refresh data (pull from the server) * Settings/Advanced - this would have additional sub-menu with following options:   + Change data source location   + Display data information (like when it was last updated, source location, etc)   + Place to specify the URL (try nanoHTTP need to add XML file-type)   + *Ability to write actuator state to the database (if time permits)* * Exit |

|  |  |
| --- | --- |
| **ID** | UI3: Main Screen |
| **Area** | UI |
| **Description** | Main screen will comprise of:   * Model of the building in the centre of the screen with appropriate sensor indicators mapped onto it * Zoom in/out buttons on the bottom of the screen |

|  |  |
| --- | --- |
| **ID** | Nav1: Navigation |
| **Area** | UI/Navigation |
| **Description** | User should be able to freely navigate in the 3D space. That means ability to view object from different perspectives (rotate around X and Y axes) and to zoom in/out on the specific point. |
| **Notes** | Rotation around Z axis should not be necessary and could complicate navigation. Also, as the model will be a building, this will provide a feeling of "stability". |

|  |  |
| --- | --- |
| **ID** | Nav2: Touch-screen navigation (rotate) |
| **Area** | UI/Navigation |
| **Description** | User will be able to navigate using the touch-screen. For instance sweeping the finger down across the screen, will rotate the model around the X-axis "towards" the screen, while sweeping a finger left will rotate the model "left" around the Y-axis. |

|  |  |
| --- | --- |
| **ID** | Nav3: Touch-screen navigation (pan) |
| **Area** | UI/Navigation |
| **Description** | User will be able to pan the view by tapping the arrow buttons located on the sides of the screen. |

|  |  |
| --- | --- |
| **ID** | Nav4: Buttons navigation |
| **Area** | UI/Navigation |
| **Description** | It will also be possible to navigate (pan) using physical direction buttons.  Pressing the middle button will restore model to its default view (isometric). |

|  |  |
| --- | --- |
| **ID** | Nav5: Zooming in/out |
| **Area** | UI/Navigation |
| **Description** | Zoom in/out ability would be provided by two on-screen buttons at the bottom of the screen (see Google Maps on Android emulator). Zoom in could also be initiated by a double tap on the screen, in which case it would zoom into the point of the tap (provided it's somewhere on the model - we don't want to zoom into an empty space) |

|  |  |
| --- | --- |
| **ID** | Nav6: Automatically adjusting view to the phone position |
| **Area** | UI/Navigation |
| **Description** | Software will display the model of the building in a "portrait" mode if the phone is held vertically and in the "landscape" mode when the phone is held horizontally. |

# 5. Work Record

### Week 1:

* An initial presentation was created with high-level ideas about how to approach the project
* Some draft, high-level requirements were included:
  + Ability to work with OWL XML files and visualise data contained in them
  + Allowing the user to freely navigate in the 3D space
  + Expandable design for additional sensors/actuators
  + Three major development areas were identified: internal database, OWL XML parser and a 3D environment

### Week 2

After preparing the presentation, which approached the project from a technical perspective, it was necessary to consider the project management side of it and choose best suited software development methodology.

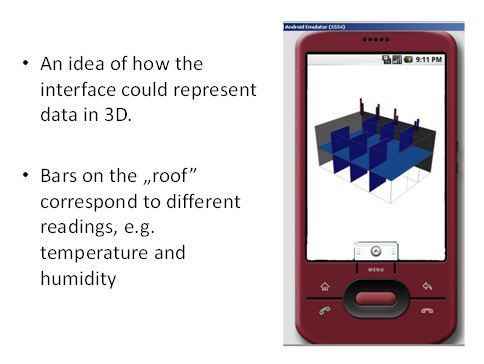
Brief research in this field showed few possible approaches. For more detailed description of chosen methodology and rationale behind it, refer to the Literature Review and Design chapters of this report.

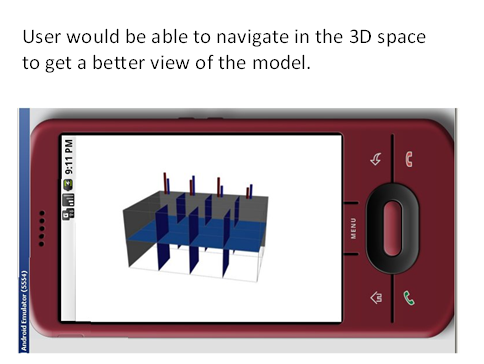
In summary, a variation of "classic waterfall" model was chosen due to its simplicity, but with few changes:

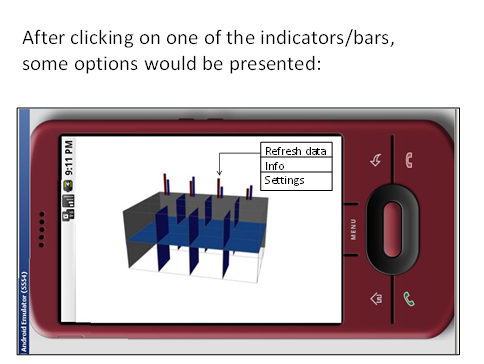
* Development stages overlap
* It is possible to re-iterate through them if necessary
* There will be few releases (probably high, medium and low priority) to ensure that at least some functionality is delivered on time.

### Week 3

The interface mock-up was prepared to clarify with the customer what the application should look and behave like. These were just slides with a concept of how the different screens in the application could look like:







Although the mock-up was not exactly what the customer had in mind, it was reasonably close and helped with reaching an agreement over the final "look and feel" of the application.

### Week 4

Few basic questions about the application in general were prepared as a starting point for creating formal requirements. These questions were asked and discussed with the customer:

- What environment is the software aimed for?

- Who is it aimed for (I need to understand who to target the interface for etc.)?

- How fast does it have to be? Do we need it to perform in real-time or on previously recorded data?

- Where will the sensory data be stored and how? How can the device connect to this repository?

- Will the sensor data be processed on the device itself, or will it be processed somewhere on a server and the device will get ready-to-visualize data?

- What types of sensors should we expect and what data will they provide?

- Where will the sensors be located? Will we have their location data?

### Week 5

There was some concern about 3D capabilities of Android (whether it will be possible to import and use a 3D model), but after some investigation it seems feasible. If it wasn't possible, different solutions could be considered, such as Adobe Flash. However, Android includes OpenGL ES 1.0, which should be more than capable of handling a building model. The problem now is to find out how to build the model on Android platform. This looks slightly complicated and substantial amount of time is anticipated to be spent researching this issue.

It was also noted, that there is a possibility to process the data (convert from OWL format to SQL or other, Android-compatible) externally, before sending it to the device. This, however, should only be treated as a contingency option and avoided if possible.

### Week 6

Draft requirements were prepared, before the next meeting with the customer. These were based on earlier-discussed questions and should be close to finalising, but are still subject to change.

Also more research on the 3D modelling and programming has been carried out to reveal the following:

* Creating the model programmatically, would be extremely time-consuming and thus not feasible - the only reasonable option is to build the model in a specialised, external application (Blender seems like a good candidate) and then import it onto the Android platform
* While building the model should be straight-forward, if not quick, finding a way to import it might present some difficulty. There seems to be no officially supported way of doing it and so far no direct instructions have been found either. It might be possible to write a Java loader (or re-use an existing one) to accomplish the task.

### Week 7

In order to interpret OWL XML files in the application, it is necessary to use a parser. Few different OWL parsers are available online:

* Jena
* Sesame
* JRDF

Since Jena is already used by researchers at the University, it was decided to go forward with it, since more resources and help would be available this way.

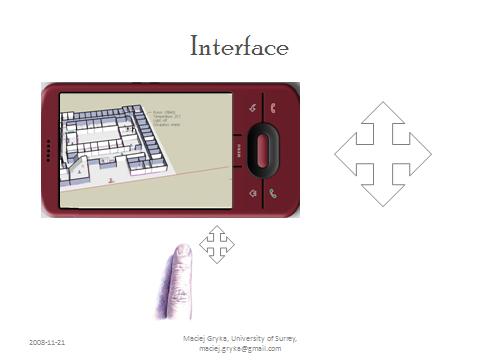
### Week 8

The formal requirements have been finalised with the customer. Please refer to the Requirements section for details.

### Week 9

The refined mock-up of the application “look-and-feel” has been produced and presented to the customer. Positive feedback was received:







# 6. Future Work

## 6.1 Development

In order for the project to be complete, the following tasks need to be completed. See Appendix 1 for the Gantt chart containing the tasks and schedule information.

* + Detailed Design
    - OWL Interpreter class definitions
    - Framework for OWL interaction, that allows reading and writing records, searching for a field within a data set (querying)
    - Define a detailed data model (classes) to represent rooms, sensors, indicators
    - Create a framework to interact with the 3D environment (change colours, lighting, textures etc.)
    - Adapt it for interpreting the required classes and representing them in the 3D environment by changing appearance
  + Implement the OWL interpreter
    - Determine which parser to use
    - Build around it to satisfy the design – add support for specific data format used in the project
  + Import the 3D model into Android
    - Learn how to import an externally-built 3D object (.md2, .obj, .3ds)
    - Document the process and troubleshooting
    - Build the 3D model of the building to be used in the project with maximum optimisation
    - Import it to Android
  + Implement 3D interaction classes as specified in the design

## 6.2 Final Report

The final report describing the entire project will be divided into following sections:

* + Introduction, as seen here.
  + Project Description, as seen here.
  + Literature Review, as seen here, with more information on similar projects.
  + Architectural and Detailed Design, describing how the system was built on the highest level as well as detailed implementation (class definitions etc.)
  + Implementation, describing how the functionality is implemented, i.e. algorithms and optimisation techniques used, documentation about how some parts were accomplished (like 3D model loading), code snippets.
  + Work Record, as seen here expanded with record of work done after the submission of this report.
  + Testing with procedures described in detail as well as test results to confirm that the project satisfies the requirements.
  + Conclusions containing project summary, both from requirements perspective and how the implementation was done, major risks that were mitigated and learning outcomes. Suggestions for future expansion (new features).

# 7. Conclusions

This report introduced the aim and initial work done for the project. Objectives and literature review were presented to show progress to date and also indication of the work to follow has been given.

The project proved challenging, mainly because it required the author to be familiar with many new concepts and work with them simultaneously (mobile development, 3D programming, OWL etc.) Also, as “cutting edge” technology is used, resources are limited to aid in research of the topic and development. However, author feels that it is an extremely rewarding experience that will prove useful in future work and research.

# References

Burns, B.B (2007), *AndroidGL* [online], Available: <http://code.google.com/p/android-gl/> [13/1/2009]

DiMarzio, J.F (2008) *Android - A Programmer's Guide* [electronic], The McGraw-Hill Companies

Dutko, A. M. (2008) ‘Reviews: Domo Arigato Mr Androidato? An introduction to the new Google mobile Linux framework, Android’, *Linux Journal* [online], vol. 2008, issue 167, Available: ACM Portal, ISSN: 1075-3583 [13/1/2008]

Eclipse Foundation (2008), *Project Summary – eclipse* [online], Available: <http://www.eclipse.org/projects/project_summary.php?projectid=eclipse> [6/1/2009]

Google (2008), *Documentation – Android*, Available: <http://code.google.com/android/documentation.html> [5/1/2009]

Haseman, C. (2008) *Android Essentials* [electronic], Berkeley: Apress

McConnell, S. (2004) *Code Complete*, 2nd edition, Redmond: Microsoft Press

McConnell, S. (1996) *Rapid Development: Taming Wild Software Schedules*, Redmond: Microsoft Press

Royce, W. W. (1987), ‘**Managing the development of large software systems: concepts and techniques’,** *Proceedings of the 9th international conference on Software Engineering* [online], Los Alamitos: IEEE Computer Society Press, pp. 328 – 338, Available: ACM Portal [13/1/2008]

suhas.gavas [forum user] (2008), *A 3d Project – Loading and Rendering md2 model :: anddev.org* [online], Available:

<http://www.anddev.org/a_3d_project_--__loading_and_rendering_md2_model-t3074.html> [18/1/2009]

World Wide Web Consortium (2004), *OWL Web Ontology Language Guide* [online], Available: <http://www.w3.org/TR/2004/REC-owl-guide-20040210/> [15/1/2009]

World Wide Web Consortium (2004), *OWL Web Ontology Language Reference* [online], Available: <http://www.w3.org/TR/2004/REC-owl-ref-20040210/> [15/1/2009]

World Wide Web Consortium (2001), *W3C Semantic Web Activity* [online], Available: <http://www.w3.org/2001/sw/> [19/1/2009]

# Appendix 1