App Double Screen

CS495 Project Report

Scott SETO
Pierre-Olivier VANHEECKHOET

CS495 – Fall 2011

Table of contents

I. Description 4
- A. Outdoor Localization 4
- B. Indoor localization 4
- C. Targeted Audience 4
- D. Double screen optimization 5

II. Motivation 5

III. Component 6
- A. Outdoor Navigation 6
  - 1. Interface 6
  - 2. Functionalities 9
  - 3. Dual Screen and Single Screen Specificities 11
- B. Indoor Navigation 12
  - 1. User Interface 12
  - 2. External Server 16

IV. Assignment 18
- A. Pierre-Olivier VANHEECKHOET 18
  - 1. Outdoor list of locations and overlays 18
  - 2. Outdoor Navigation Interface 19
  - 3. Outdoor path 20
  - 4. Accelerometer 21
- B. Scott SETO 22
  - 1. Indoor Navigation 22

V. Lesson Learned 28
- A. Dual Screen emulator and Google APIs 28
- B. Overlays on the real dual screen device 28
- C. Problem to launch emulators on eclipse 28
- D. Problem with the change of orientation on the dual screen phone 28
- E. Problem with GPS service 29
- F. Finding Unique Access Points in a Building 29

G. Variances in Accelerometer Readings 29
H. Using a Dual Screen Phone 29
VI. Future Work 30
A. Create More Floor Plans 30
B. Add More Information about Rooms and Classes 30
C. Use the Accelerometer to Gain Wi-Fi Data 30
D. Add More Icons 30
E. Work on Both Single and Double Screen Phones 30
F. Add Custom Places 31
VII. References 31
A Star Algorithm For Path Planning 31
VIII. Appendix 32
A. Orientation simulator 3-Axis 32
1. Phone orientation model 1 and bases change figures 32
2. Alternative Phone orientation model (same results) 33
3. Bases changes 1/3 33
4. Bases Changes 2/3 34
5. Bases changes 3/3 34

Table of Figures

Figure 1 : Outdoor Interface, Full Landscape mode 7
Figure 2 : Outdoor Interface Layout Structure (all the names refer to a Linear Layout) 7
Figure 3 : Menu Bar 8
Figure 4 : Types of locations and corresponding icons 10
Figure 5 : Full-portrait and Full-Landscape modes 11
Figure 6 : Single screen portrait and landscape modes 11
Figure 7 : Room selection menu 12
Figure 8 : Indoor map with path 13
Figure 9 : Indoor map with icons 14
Figure 10 : Context menu 15
Figure 11 : For Outdoor list and overlays and downloading from a server 19
Figure 12 : Indoor map with walking area 23
Figure 13 : Indoor area with grid 24

I. Description

Our application has two goals: provide students and university staff outdoor and indoor location by using many localization systems and provide instructions to go to a certain location. The application must be able to provide a precise location in multiple environments by using the available localization systems and display the user's location on a map. If a localization system is not available the application will use another kind of system in order to find the user's location in any type of environment. The user can navigate on the map at the same time that he is able to obtain driving or path instructions.

A. Outdoor Localization

For the outdoor localization system the application uses the GPS to find the user's location. Itis able to display a list of locations around the user and provide driving instructions to them.

The user can select a location on the list or on the map. The user can read the name of the location, the distance between its current location and the chosen location and also see the position on the map. The user can also obtain driving instructions to a precise place. In this case the application draws a path on the map and shows driving instructions. The user can continue to navigate and explore the map after a location is chosen as destination. When the user wants to find a location inside a certain building the outdoor system launch the indoor system in order to guide the user through the building.

B. Indoor localization

For the indoor localization system the application display a map of the building and a list of specific locations like rooms, lifts, stairs… When the user chooses to go in a certain part of the building it provides a path to the chosen location. The application is able to control multi-floors plans and lifts and stairs as ways to go to another floor. The application can't use GPS in an indoor environment so it uses other sensors like Wi-Fi, accelerometer and camera and QR-codes to locate the user and display its location on the map.

C. Targeted Audience

Students and professors, staff, guests are our main targeted audience. Students and professors often need to find a room quickly, especially when they have not a lot of time between two classes. This application could be also useful for guests because they often need to find the place of a conference or presentation.

D. Double screen optimization

The application runs on a double screen phone: Kyocera Echo. Thus it is adapted and optimized for a use with two screens. The main advantage of the double screen is that it allows separating completely the controls and the view. In the case of our application the controls consist of the choice of a location, the driving instructions, the change of the floor and the view consist of the map. With a double screen the application is able to provide rich controls on a screen and always display the map and the user location at the same time on the other screen. Thus the user doesn't have to often interact with the interface and the map to have a good global view of its location and the surrounding points of interest.

II. Motivation

The main motivation for the project comes from a lack of GPS range and accuracy indoors. The GPS has a hard time penetrating the ceilings of buildings with multiple floors or with a lot of reflective materials. It is necessary then to have a positioning system that works indoors with good accuracy and not a lot of setup.

Buildings can be very large and it can be difficult to navigate them. It may not be intuitively obvious how to get to a certain room in a building. Many buildings do not have indoor maps where you can find where you are going. Also, most maps found in buildings are not mobile and you can't have access to them wherever you want. If you need to know the shortest path to your destination, it is not always obvious.

When GPS and Wi-Fi are not available, there must be a way to navigate a building and find our way around it. Especially if a room has narrow hallways and small rooms, it is hard for an application to find where you are. Being able to use the accelerometer to find a change in position is very helpful in determining your location.

Also, when a user wants to find a specific area in a room that is not physically marked, the user will benefit from a map that shows the locations of important areas in a building. Then, the user will know where they are and how to get there. For example, if there is a computer lab in a certain building and the user wants to find it, they can look at an indoor map on a phone and find where it is.

III. Component

A. Outdoor Navigation

The Outdoor Navigation provides the location of the user on a map, a list of the surrounding points of interest and also a path and driving directions to a specific location.

1. Interface

The interface for the outdoor navigation is optimized for double screen phones but can also work with a single screen phone. The interface is based on a custom tab view that allows displaying only interesting data on the main part of a screen and allows switching between different kinds of information. An example of the interface is represented on figure below. The interface is in a double screen landscape mode (Full), normally the picture is divided vertically on the center.

a) Custom Tab View

We planned to use the TabView provided by the SDK. However the Tab View displays one **activity** per tab and the tab bar fill the **full width** of the screen. Such a structure was not really adapted to our application. Indeed, in order to optimize the application for double screen phones we decided to always show the map on an entire screen (**Map Screen**) and to change the content of the other screen (**Control Screen**) in function of the information required by user. The creation of a Tab View on a screen and a Map View on the other screen wasn't easy to create; it is why we decided to create a **custom tab view**running in a single activity and based on several levels of linear layout. The structure is presented in figure 2.

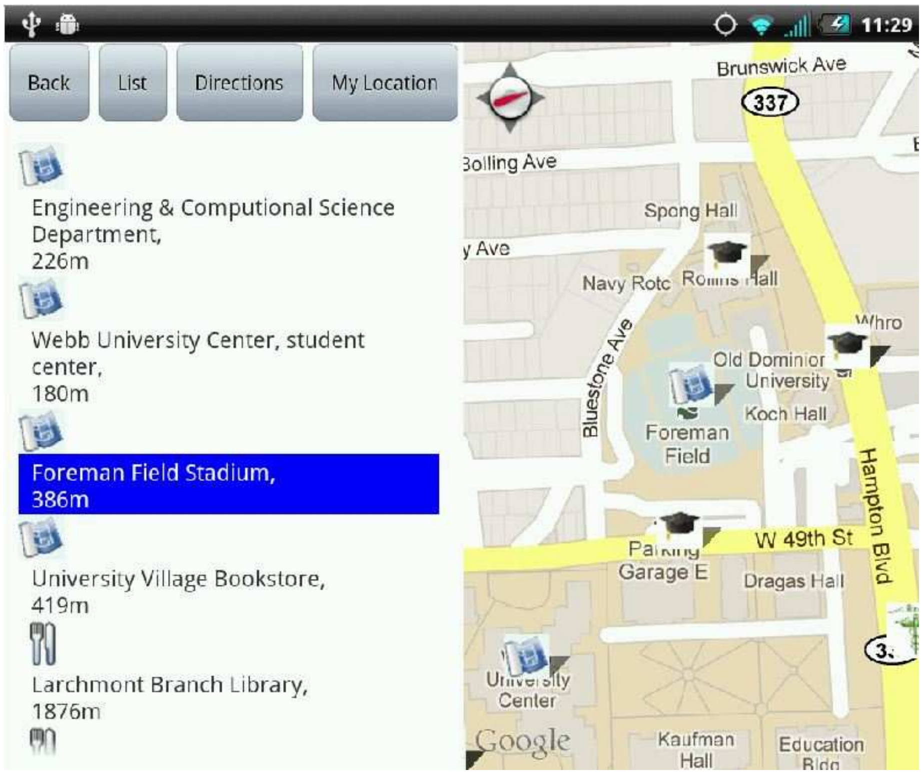


Figure 1 : 2.A.1: Outdoor Interface, Full Landscape mode

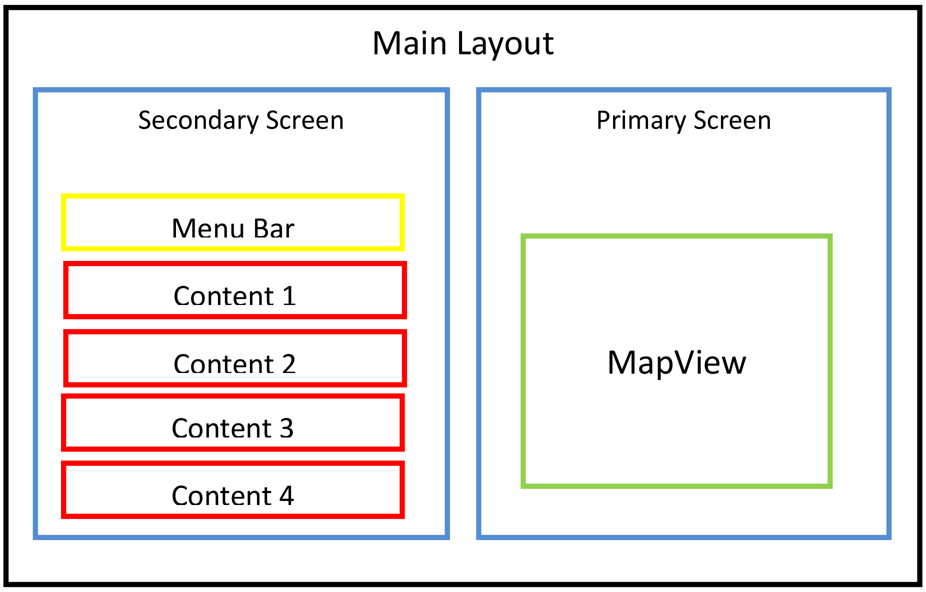


Figure 2 : Outdoor Interface Layout Structure (all the names refer to a Linear Layout)

b) Menu Bar

The menu bar is the main control part of the interface. From this bar the user can decide what information is displayed on one of the Control Screen. The Menu bar is presented in figure 3.

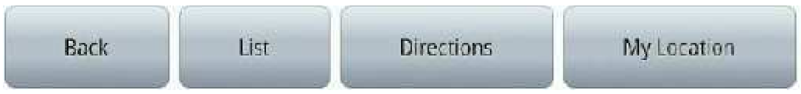


Figure 3 : Menu Bar

The Menu Bar consists of a Linear Layout oriented horizontally with four Buttons. Each button controls either the displaying of a content layout or another kind of action. The functions of the buttons are described below:

-	Back	:returns to the last indoor activity
-	List	:shows the list of the surrounding locations
-	Directions	: Display driving instructions and path
-	My location	:Focuses the map on your current location

c) Contents

In order to display different kind of information the application uses several linear layouts. Each layout corresponds to a specific kind of information. At the beginning of the development of the application it was planned to display four types of content: a list of surrounding locations, driving directions, search location and location information; finally the number of content was reduced to only the two main contents:

- A list of the surrounding locations
- A list of driving instructions

The buttons List and Directions of the Menu bar displayed respectively the surrounding locations and the driving instructions. When content is displayed its visibility is turned on View.VISIBLE and the visibility of the other content is turned on View.GONE. Thus only one content layout fills all the space below the Menu bar. This behavior is similar to a Tab View behavior.

2. Functionalities

a) List of the surrounding locations

The list of surrounding locations is generated automatically after the launch of the application. The application gets the current location of the user from a GPS service launched and running in background that update regularly the current location.The current location is sent to an external server through an URL. This URL contains the current latitude and longitude and the type of the location that is looked for. The server uses these data to send a request to the Google maps service and create a file with the received data. This file is named with the word "places" and the current latitude plus ".htm". Then the file is downloaded by the application and analyzed. The reasons of the processes of getting the locations are described in the section Lesson Learned.

After getting the file the application reads it and extracts the name, the latitude and the longitude of the location and adds the type in order to filla database of locations. Then this database is used to create Text Views that include the name and the distance of the locations from the user and that fill the content layout dedicated for the surrounding locations.

There are 6 types of locations that are the most useful for students and professors:

- Locations with indoor maps (these locations are stored directly in the external server)
- Food (restaurants, food shops…)
- Pharmacy
- University
- School
- Library

When the user select a Text View on the list the map zooms automatically on the corresponding location and the Text View is highlighted in blue.

b) Driving instructions

The driving instructions are generated automatically after the user selected a location in the list or on the map and clicked on the "Directions" button.

First the application draws a path on the map from the user's current location to the location selected. The path is drawn on the map by using a code found on "http://code.google.com/p/j2memaprouteprovider/".Secondly the application sends the current latitude and longitude and the destination latitude and longitude to the external server.

Then the external server send a request to Google maps and send the received data to the application after formatted them. Then the background of the destination location is set to red in the list of locations. It makes possible for the user to continue to interact with the list without losing the description of the destination location.

Finally the application displays the driving instructions directly in a Web View in the layout content dedicated to the driving directions. The fact of displaying the instructions directly in a Web View makes the application easier to use because we don't need to create special Text Views for the instructions. This way is not good for the list of the locations because the user needs to be able to interact specifically with each location on the list as we will see below.

c) Map View

The Map View is displayed on the second screen. Thus it is always displayed when the user uses the application on double-screen(Full) mode.The map view shows the current location of the user and the surrounding locations around him (until 2 000meters).Each type of locations is added as a new overlay on the map and each type of location is represented by a different icon as showed in the table in figure 4.

Type	Buildings with indoor maps	food	university	school	pharmacy	library
Icon	🗺	🍴	🎓	📐	⚕	📚

Figure 4 : Types of locations and corresponding icons

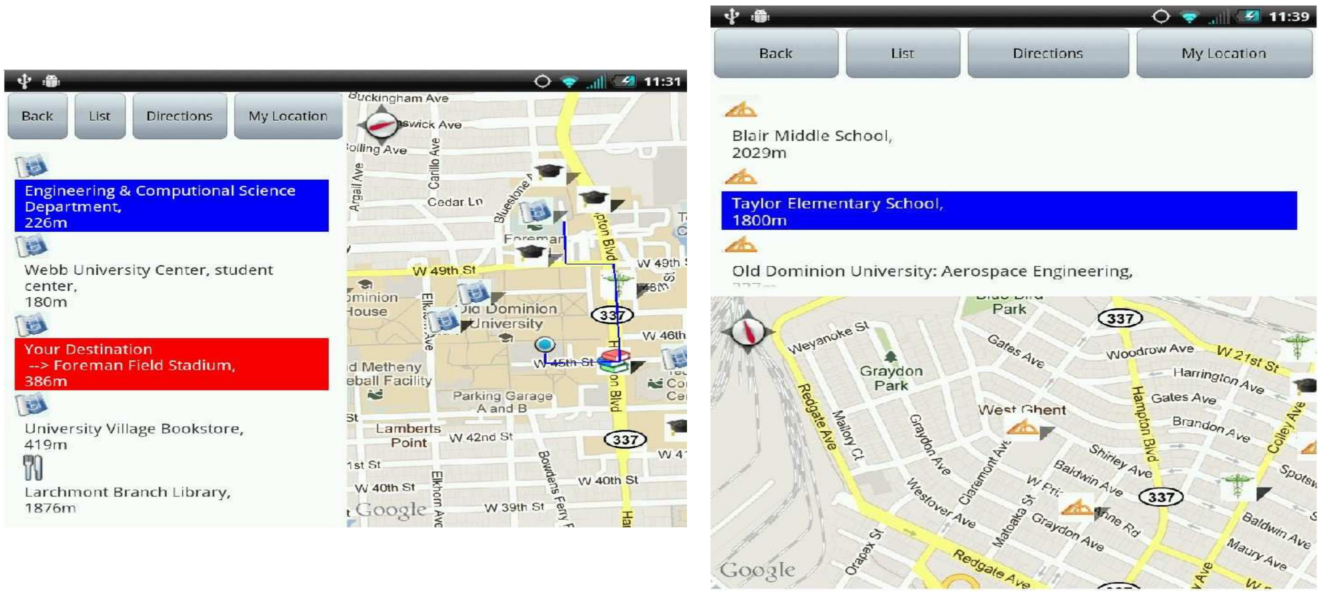
When the user clicks on a location on the map the map zooms automatically on this location and the corresponding item in the list is highlighted in blue.

d) Back to the last indoor map

The application is able to record the name of the last building you explored through the indoor navigation system and allows you to go back to the corresponding plan through the Back button. If you didn't explored an indoor map the back button simply finishes the application.

3. Dual Screen and Single Screen Specificities

a) Dual screen mode interface



Engineering & Computional Science Department, 226m

Webb University Center, student center, 180m

Your Destination --> Foreman Field Stadium, 386m

University Village Bookstore, 419m

Larchmont Branch Library, 1876m

Blair Middle School, 2029m

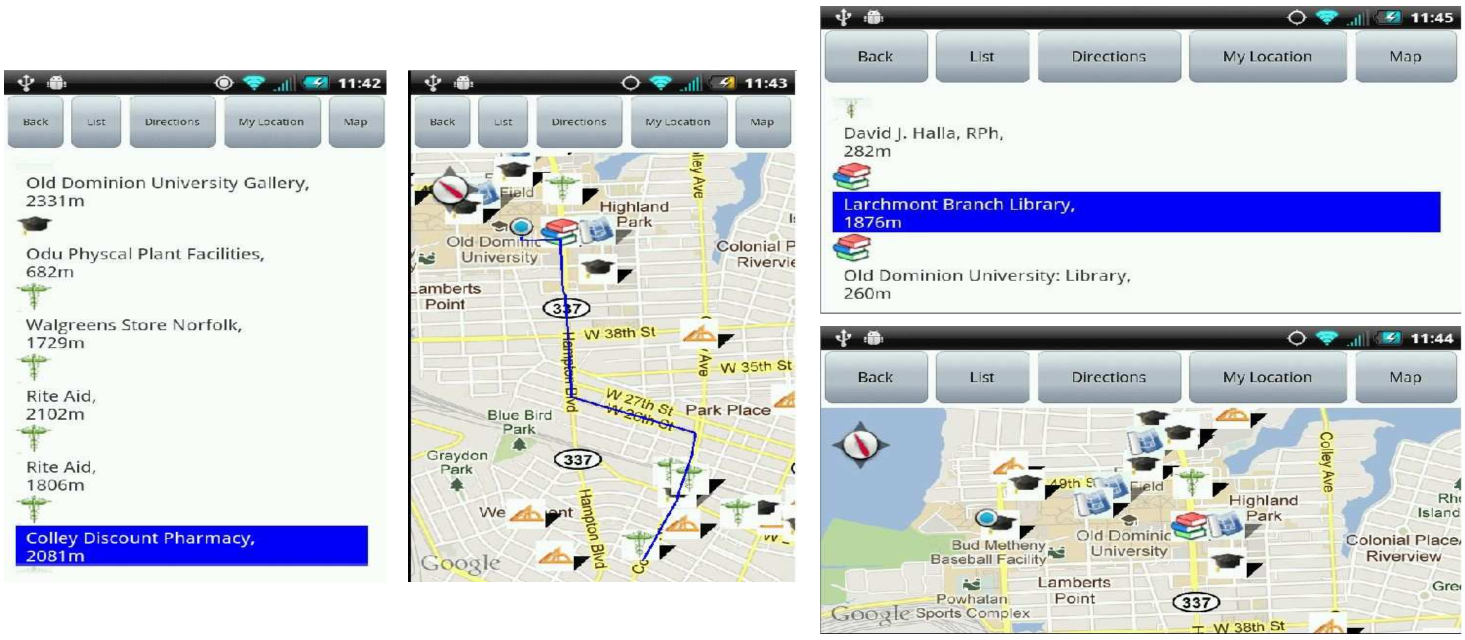
Taylor Elementary School, 1800m

Old Dominion University: Aerospace Engineering,

Figure 5 : Full-portrait and Full-Landscape modes

b) Single screen mode interface

For the single screen mode a new content layout are and a new button are added to display the map on the same screen than the other contents.



Old Dominion University Gallery, 2331m

Odu Physcal Plant Facilities, 682m

Walgreens Store Norfolk, 1729m

Rite Aid, 2102m

Rite Aid, 1806m

Colley Discount Pharmacy, 2081m

David J. Halla, RPh, 282m

Larchmont Branch Library, 1876m

Old Dominion University: Library, 260m

Figure 6 : Single screen portrait and landscape modes

B. Indoor Navigation

1. User Interface

a) Overview

The user interface has two parts to it. The first part is the view of the indoor map. It is not stationary. It can be moved around by touching the image and dragging it around. It must stay the same size and cannot be zoomed in and out. It contains various map markers that show important locations on the map. It just shows one floor. To change the floor, you use the second half of the user interface.

The second half of the user interface has the buttons to control the map. There are four different buttons. In order from the left, they are 'Change Floor', 'Go to Room', 'Draw Path', and 'Go Outside'. The 'Change Floor' button opens up a dialog box. It lets you select which floor you want to see. After selecting a floor, the image then changes to the floor that was chosen.

The 'Go To Room' button opens up a dialog box with the list of rooms in the current building. After a room is chosen from the list, the application automatically goes to the floor that the building is on and centers itself on that room.

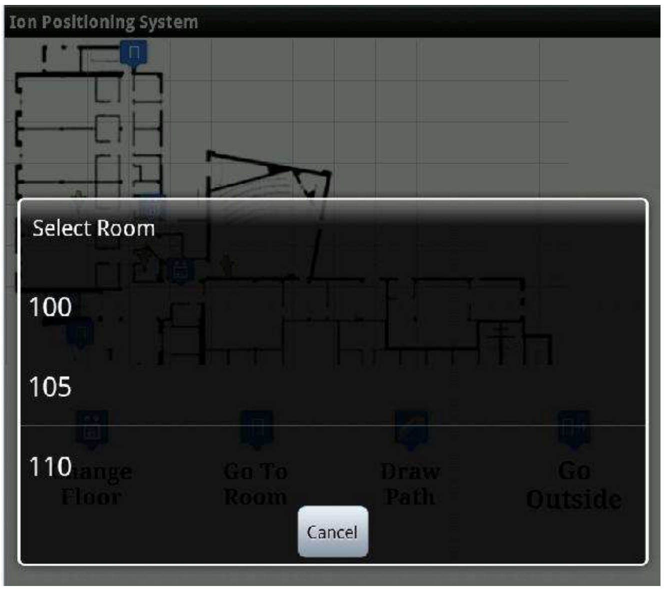
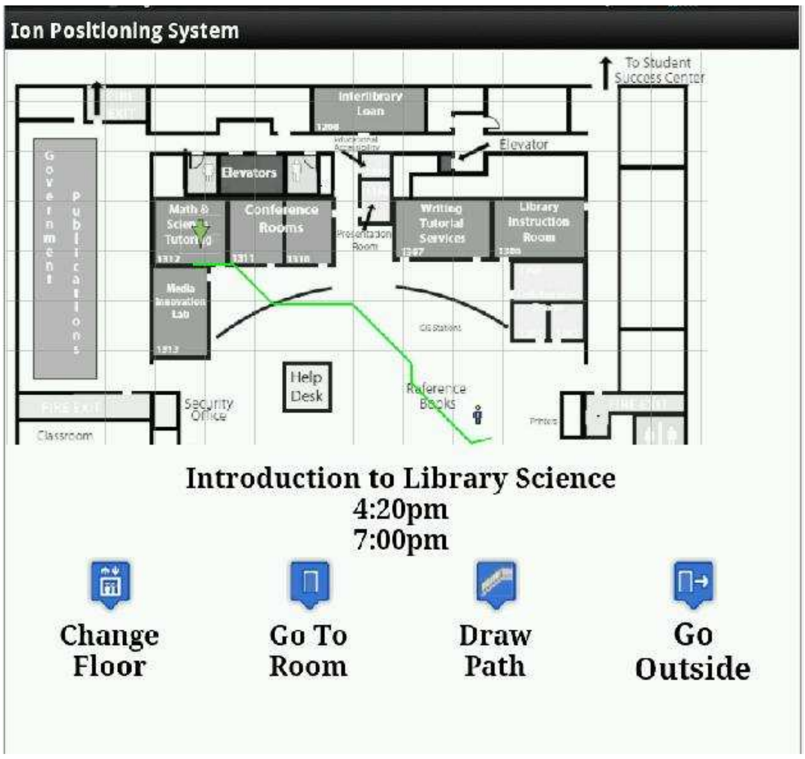


Figure 7 : Room selection menu

The 'Draw Path' button also opens up a dialog box with the list of rooms in the current building. After a room is chosen, there are two choices. If the room is on the same floor, a path is drawn from your current location to the room. If the room is on a different floor, a path is drawn to the nearest elevator or stairs. Then when you change to the floor that the room is on, a path will be drawn from that same elevator or stairwell to the room that you want to go to.



Introduction to Library Science
4:20pm
7:00pm

Figure 8 : Indoor map with path

The 'Go Outside' button lets the user go back to the outdoor part of the application. The outdoor map is then centered on the location of the building that you just left.

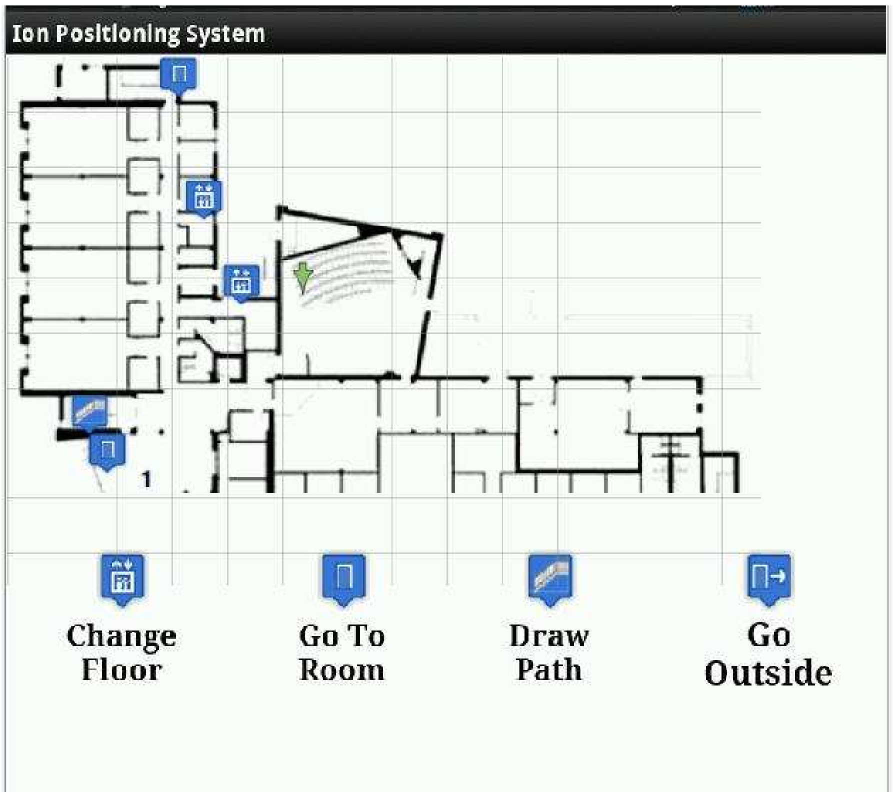


Figure 9 : Indoor map with icons

b) The Custom Display View

The application uses a custom image view to display the map on the top half of the screen. The custom image view is used to detect presses and long presses on the image. It is also used to detect when the image is moved and the distance that it is moved. This information is important because it allows the application to respond to user interaction with the map. For example, it can show room information, or collect Wi-Fi data, or update your location based on a certain press on the image.

The custom ImageView contains a custom Drawable. This custom Drawable allows the application to control the drawing of a path or icons onto the custom ImageView. Each Drawable contains a Bitmap that holds the image for a certain floor. When the custom display view is set up, the custom Drawable is manipulated to display map markers or paths based on information retrieved from a database or interactions from the user.

c) Map Markers

Each building floor map contains markers that show important areas on the map. The markers that have been created show the entrance, exit, elevator, and stairs. Each marker is a specific Bitmap image that is overlaid onto the custom ImageView. The locations of these markers on the map are retrieved from a database. They cannot be changed from inside the application. Some of the markers are used in a specific way by the application. When a building is first entered before a location is known. The path to a room starts from the location of the entrance marker. When a path is to be drawn to a different floor, the path first goes to the location of the closest elevator marker.

d) Switching Between One and Two Screens

The application works with both one and two screens. If running on a dual screen phone, the application can work with the phone folded up or opened to two screens. If there are two screens available, the map is placed on the top screen and the buttons to control the map are placed on the bottom screen. If there is just one screen, the map is still placed on the screen, but the buttons are accessed by pressing the menu button the phone itself.

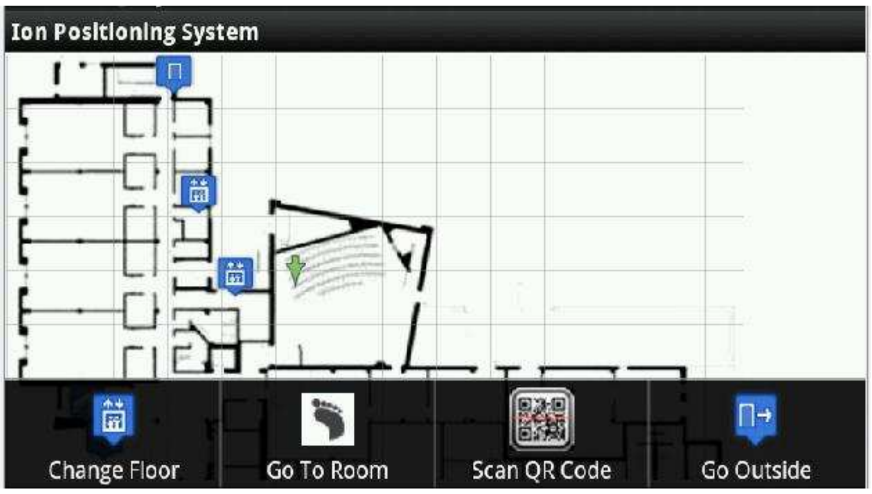


Figure 10 : Context menu

e) Switching Between Indoor and Outdoor

When the user wants to switch between the indoor to the outdoor, the user must press the ‘Go Outside’ button on the bottom screen on the user interface. If the user is in the outdoor activity, the user must click on a specific building on the outdoor map. Once clicked, the application switches from the outdoor activity to the indoor activity. If no location is found, the application displays the first floor of the building. If a location is found, the application displays the floor that the user is on and the location within the building.

2. External Server

a) Overview

There is an external server running Apache 2.2, PHP, and MySQL that is servicing the application. There are various PHP scripts that are used to get data from the MySQL database and to send it to the application. The external server stores data about indoor maps, map markers, and Wi-Fi localization data. The data is shared among all instances of the application that may be running on different phones at the same time.

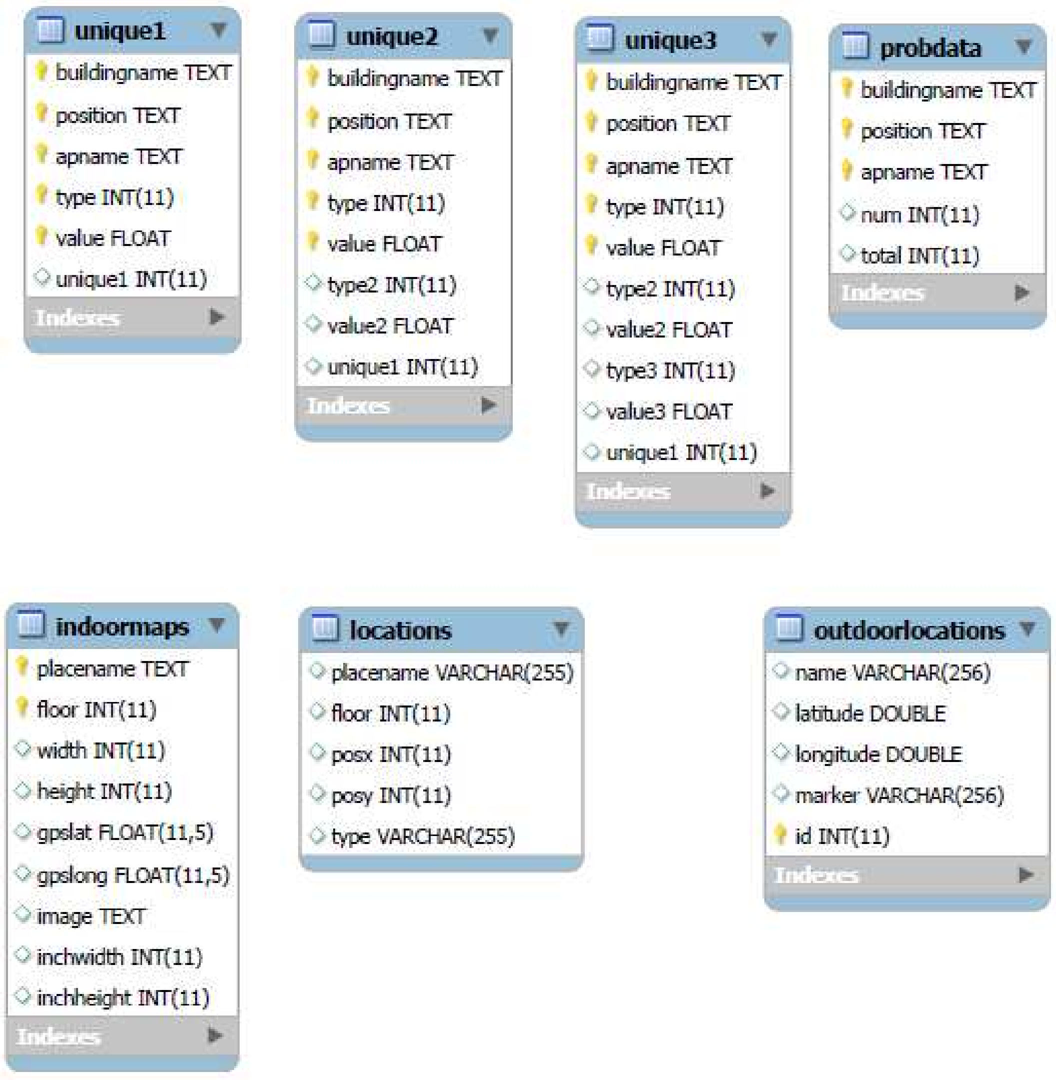


Figure 11 : Database Diagram

b) Database Interaction

There is database interaction between the external database and the application in the following ways. The PHP scripts on the server are able to convert tables on the external database into SQLite INSERT statements used to transport the data on the external server to tables on the internal database. An internal SQLite database is used on the phone and the table structures mirror those on the external server. When the application starts up, the PHP scripts are read from by the application, the internal database is cleared and the data from the external database is imported into the internal database.

c) Using the External Database

The external database is not only used to copy the data over to the internal database. It is also used to save Wi-Fi access point information into the database. Whenever Wi-Fi access point information is read by the application, the building, the floor, and the various access point measures are stored. Also, the database can be edited to evoke change in the application. For example, map markers can be added or their position can be changed by editing the database. Also, room information can be added. There are XML documents stored on the external server that contain information about rooms in different buildings. These XML documents can be edited directly on the server to change class information.

d) Using the Internal Database

The internal database is used to hold information that is necessary for the application to run properly. Its purpose is to make necessary data immediately accessible from the application. It is able to convert data that is stored in the database and convert it to objects that can be used effectively within the application. For example, the application creates a Location class and a Map class used to hold information about map locations and information about map images. Instead of going to the database every time information is needed, the application goes to an ArrayList of Location or Map objects to get the information that it needs.

e) Data Storage

The database stores the indoor maps, room data, and map marker data. It is stored in various ways. Indoor maps are stored as drawables in either JPG or PNG format. The database stores size information about the maps. It contains the pixel width, pixel height, width in inches, and height in inches. The database also stores the name of the indoor map. Room data is stored in XML files on the server. The application used XPath to extract the information from the files and then the information is written into an application object that stores the data. Map markers are stored by their grid coordinate, the type, the building name, and the floor that they are on.

f) Saving Wi-Fi Data

Wi-Fi data is stored differently from other data in the application. Each Wi-Fi access point has a MAC address that distinguishes it from other Wi-Fi access points. This address is stored in the database. Also stored is the building name, the floor, and different measurements derived from the SSID. Just one SSID is not stored. The application takes 7-10 readings the Wi-Fi data. It tabulates the minimum, the maximum, mean, and median of the SSIDs that were read. These values are then stored in the database with the specific MAC address, building name, and floor.

IV. Assignment

A. Pierre-Olivier VANHEECKHOET

Personally I worked on five main parts:

- Outdoor navigation interface
- Outdoor list of locations and overlays
- Outdoor Path
- Accelerometer

At the beginning of the project we could not have access to a dual screen phone so we were working on single screen phone or on the emulator. Thus a lot of first tests and test-applications were not making for dual screen for the beginning.

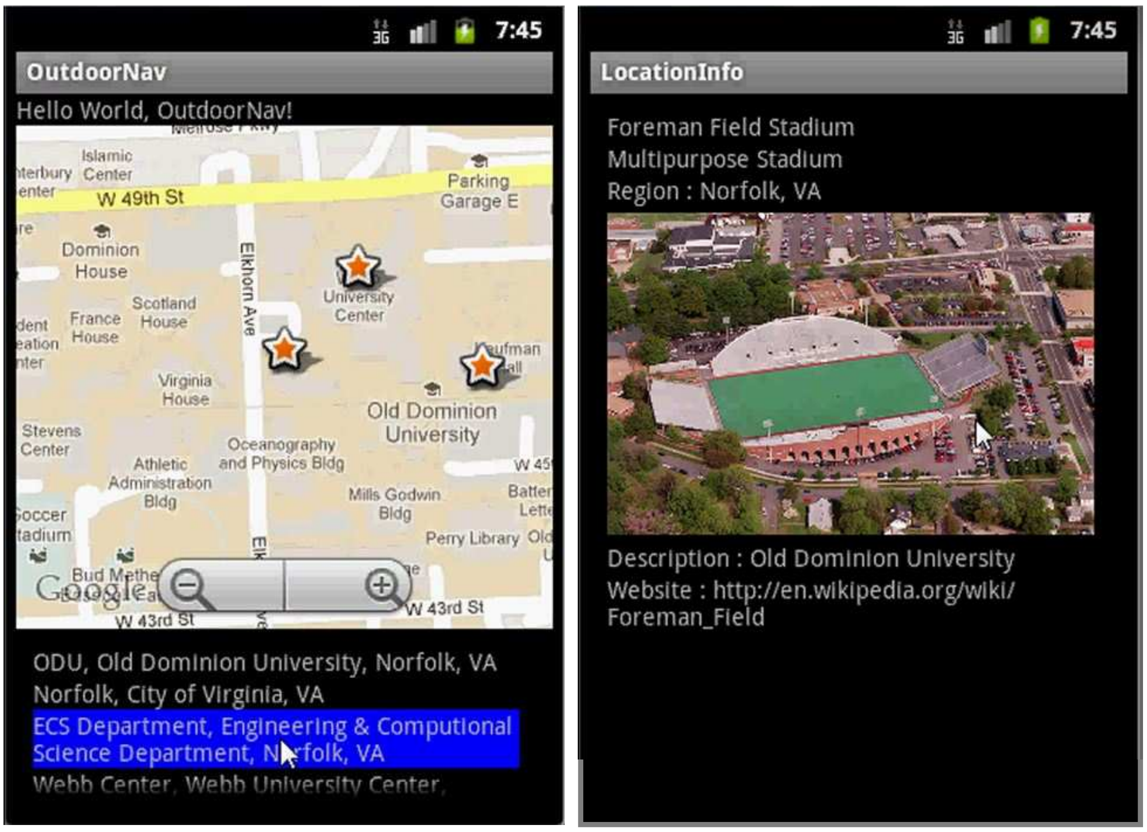
1. Outdoor list of locations and overlays

Before doing a version fully dedicated to dual screen phone of the Outdoor applications I created a test-application on a single screen phone emulator with a simple interface in order to test the possibility to download locations from a server and update the overlays on the map.

Thus I created a simple text file containing several kinds of data about a location like the latitude, the longitude, the name, the description, a URL to a picture, a URL to a site web and even an URL to a file describing related locations. After I put it on a server (my own pc) and tested the application.

This application downloaded the file and extracted the corresponding locations. Then it created a scroll list of Text Views and an overlay with a location corresponding to a specific Text View and a specific item on the overlay. When I clicked on the list the map zoomed automatically on the location. When I clicked on the icon on the map the application either displayed information about the location like picture, description, and functional hypertext link to a website or, if the corresponding item had a URL to another file of locations it downloaded the file and added the new locations on the map and in the list. The application is presented in figure 3.A.1.

But in this application the list and the map don’t fully interact and it is not possible to select an item in the list by selecting the corresponding point on the map. The data were also stored in text files on the server and not in databases. So during the next works the interaction between the map and list, the control of databases and the addition of driving instructions and the drawing of a path were adding. All of these additions were affected by the use of a double screen phone during the rest of the development period and I had to create a specific interface able to control both double and single screen mode.



ODU, Old Dominion University, Norfolk, VA
Norfolk, City of Virginia, VA
ECS Department, Engineering & Computional Science Department, Norfolk, VA
Webb Center, Webb University Center,

Foreman Field Stadium
Multipurpose Stadium
Region : Norfolk, VA

Description : Old Dominion University
Website : http://en.wikipedia.org/wiki/Foreman_Field

Figure 12 : For Outdoor list and overlays and downloading from a server

2. Outdoor Navigation Interface

One of the main features of this application is that it is made to runs on a dual screen phone and to benefit from its specificity.

One of the great advantages of a dual screen phone is that it allows to separate a view (here a map) from the controls (here the list of the names and distances of the locations) or to share the out between the two screen and thus lighten the interface and reduce user’s actions. In order to benefit from these characteristics I created an interface that separates the map from the contents controls.

I also created a full interaction by linking the items of the list of locations with the icon of the overlays of the Map View. When the user select an item in the list the map automatically zooms on this location and the Text View is highlighted in blue. This allows the user to find the location of a building from its name. I also implemented the reverse action: when the user clicks on an icon on the map the map zooms on it and the corresponding Text View in the list is highlighted in blue. This allows the user to find the name of a building from its location on the map.

The link between the Text Views and the overlays is based on the ID of the location. This ID is the ID of the location in the database. I add it inside the Text Views by using "setTAG()" that allows me to set a string value in a Text View. I add the ID inside the overlay items by using the "snippet" value that allows me to set a string value in an overlay item. Thus I can identify which icon and which Text View corresponds to which locations.

For the driving instructions and the surrounding locations I wrote the parts of codes that allow the application to send URL receive and understand data from the web. My partner did the code in server-side thanks to its greater knowledge in this domain.

3. Outdoor path

I implemented the drawing of the path from the current location of the user to his location. I looked for an existing API for the Map View but I found nothing about path. I looked for an explanation or a part of a program that could help me in my task and I found a code specially done for this. This code comes from the following URL:"http://code.google.com/p/j2memaprouteprovider/".I modified the code in order to implement it in the application. After several try it works well.However the drawing of the path is dependent of the knowledge of the update of thecurrent location of the user.

4. Accelerometer

In this application we wanted to implement a localization system that could work without any kind of wireless signal. We thought about the accelerometer and the gyroscope.

The gyroscope can give us the movement undergoes by the phone when it moves but we didn't receive data from it. We received data from the orientation sensor and the accelerometer but never from the gyroscope. We don't know if there is a gyroscope in the double screen phone.

To avoid this problem I created a program that is supposed to provide the acceleration on the three axes of the accelerometer when the phone is moving. But to realize this measure I had to cancel the terrestrial gravity: "$\vec{g}$". Indeed the orientation of the axes of the accelerometer is linked to the orientation of the phone. Thus, when the phone is on the table, only the z axe has a value of acceleration different from zero. But when we turn the phone $\vec{g}$ is projected on two or three axes and it becomes difficult to determine the acceleration of the phone caused by a movement.

I looked for a model to cancel the terrestrial gravity but I didn't find exactly what I was looking for. So I decided to use my knowledge in mechanic (I studied mechanics during my second year of university) to create a model of the orientation of the phone.

I decided to use the orientation sensor to know the orientation of the phone in comparison to earth and the north because the orientation sensor uses the compass. It took me a lot of time to create all the equations for the model and to implement them.. The model and the formulas are presented in the appendix A: Orientation simulator 3-Axis.

When I tested it I obtain good result for rotation over the ground but never for other rotations of the phone. I checked my formulas two times and never found an error. I finished by noticing that one of the angle provided by the orientation sensor doesn't go from 0 to 360 or -180 to 180 but only from -90 to 90. Consequently the value of this angle is the same for two different orientation of the phone. I spend a lot of time to check formulas, looking for better model and implanting formulas so I finished by leaving it aside and working in another way to use the accelerometer to know the movement.

My second model consists of using the accelerometer to create a step counter and to use the compass to know the direction of the movement of the user.

I used four parameters in order to determine if the user did a step or not:

- the absolute value of the total acceleration
- the interval between two measures
- a threshold for the acceleration
- the duration of the acceleration above the threshold

When the user does a step the absolute value of the acceleration goes over the threshold and stays above it during a certain period of time. If this period corresponds to the duration of a step, a step is counted.

It was not easy to calibrate the pedometer. It worked pretty well when Ididn't use the duration of the acceleration above the threshold but wrong movements are also counted. We tried to minimize the impact of the wrong movements by usingthe duration of the acceleration above the threshold but it makes the pedometer less accurate.

We decided to use my pedometer and another pedometer found through our researchesto avoid false positive. The other pedometer counts more steps than ours but not always the same than ours. By using the two pedometers and comparing the results we hope to avoid some false positive detection.

Finally we did a trade-off between accuracy and resistance to the wrong movements and false positives.

B. Scott SETO

Personally, I worked on several main parts. First, I set up the folders and the database on the external server. Then, I worked on creating the places and directions information. Then I was able to create the indoor navigation application.

For the outdoor application, we needed places and direction data. I was able to learn and communicate the Google Places API and other Google classes to retrieve the directions. In the outdoor application, we needed to read the places around us and the current directions. I created a PHP script that created a file on the server that had the places information and the directions. The outdoor application just had to read the files to get the information. It was just a list of raw information that the application had to format for the application. The files took in the GPS coordinates of the current location, the start location, and the end destination in order to return a result. I worked with the Google Maps Javascript API to generate a Google Map. Using that map, I was able to extract the places and directions information.

For the indoor navigation application, I was able to display indoor maps, paths, and locations. I was also able to work with the outdoor activity. So, when the user pressed on a building, the indoor activity would know where the user is and be able to take control of the application from there. The application details are explained in detail more below.

1. Indoor Navigation

a) Overview

The following is a basic description of how indoor navigation is done within the application. It also includes more in-depth information about the algorithms that you used in the activity. It covers the algorithms that draw the path, the importance of map overlays, how map overlays are created, the use of a grid overlay, and how the application finds your position.

b) Creation and Usage of Indoor Map Overlay

In order to navigate an indoor map, the application needs to know the areas on the map that can be walked and those that are inside the walls of the building. First, a map of the floor plan for a room was loaded into an image editor. Then rectangles of a certain color were drawn on the floor plan. These denoted the areas on the map that could be walked on.

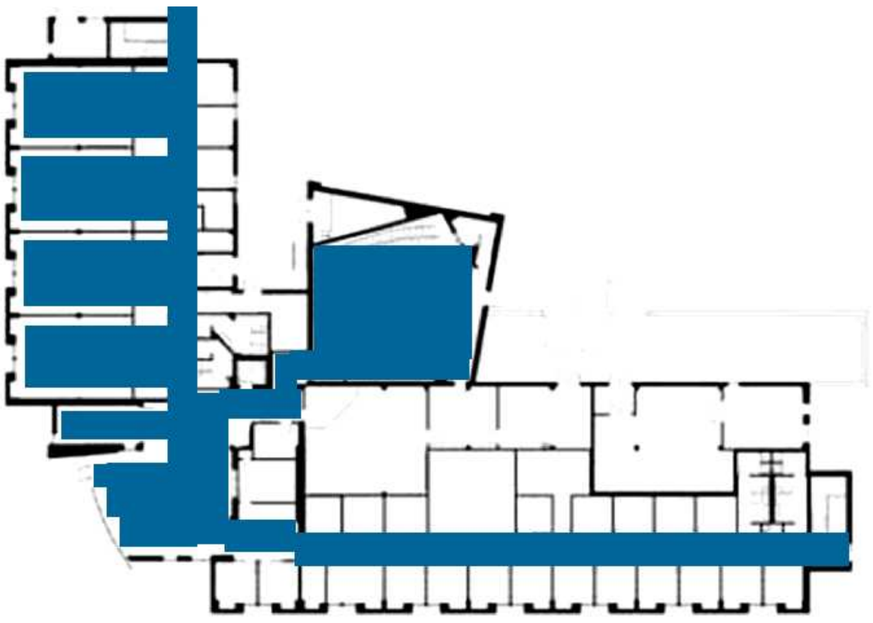


Figure 13 : Indoor map with walking area

An external Java application was created to read the image. The rest of the image was black and white, so the application could differentiate between the color of the rectangles and the rest of the map. The application then is then able to take the sections of the map covered in the distinguishing color and convert them to grids that are 50 pixels wide by 50 pixels high. The application is set so that if a certain percentage (like 70%) of the grid is covered by the distinguishing color, then it counts as a grid section. Each grid section is then identified by its position. Its position is the x and y pixel coordinate of the center of the grid space. Then the grid segments are stored in a file.

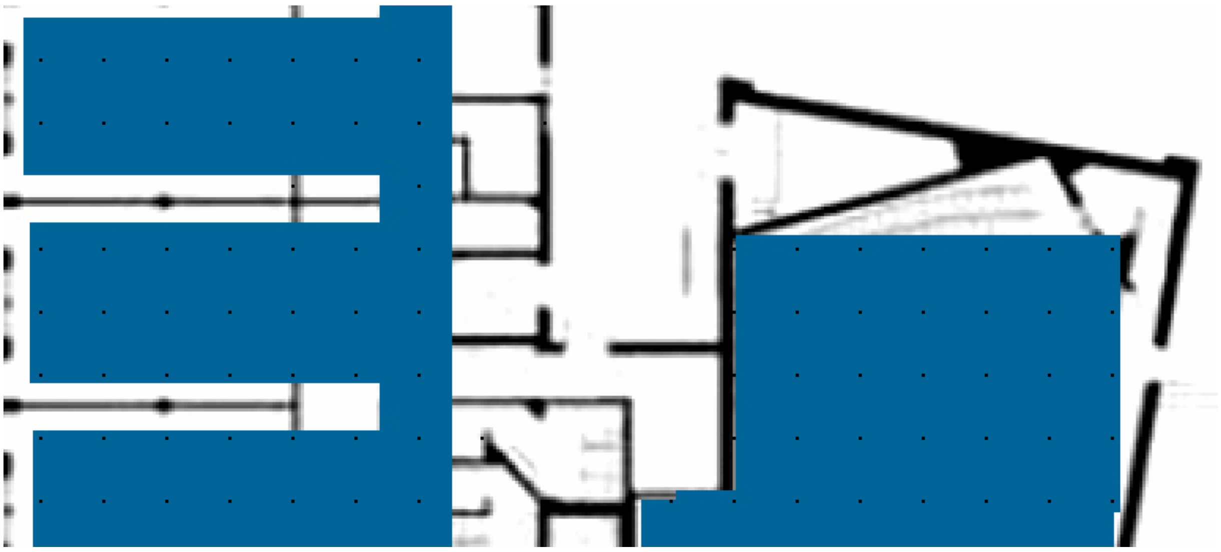


Figure 14 : Indoor area with grid

The file is put into the assets folder of the application. When the indoor application starts up, it reads this file. Using this file, a shortest path algorithm can be used from one grid space to another in order to find the shortest path.

c) Finding the Path

There are various ways to find a path from one location to another. There is an optimal algorithm called Djikstra's shortest path algorithm. There are also various heuristics. One of the heuristics is called the A* algorithm. This is the one that was used to find the path to the destination. The application has a certain set of grid segments that a person can walk on in the building. The algorithm needs to know which grid segment is adjacent to another grid segments. This is known using the pixel coordinate of each grid segment. If the grid segment is 50 pixels away and just to the left, right, above, or below another grid segment, then the two are adjacent. Also, if a grid segment is 50 pixels north and 50 pixels east, for example, then the two grid segments are also adjacent. All these adjacencies are stored in a database object called NavGraph. Then with this graph, the A* shortest path algorithm can be used. It returns the grid segments needed to go from Point A, to Point B. Then various line segments can be drawn between the returned grid segments to draw the path.

d) Building the Grid

In order to interact with the application and to calculate path navigation with only the necessary number of points, a grid is necessary for the application. It is drawn on top of the custom ImageView. Each line is drawn every 50 pixels both horizontally and vertically. The grid is then stored in the application. For example, the top left grid space has a position of '0 0' and the one to the right of it has a position of '1 0'. Then space below it has a position of '0 1'.

e) Automatic Location Correction

Every time the user moves, the application recalculates the position of the user using the Wi-Fi signals. For 7-10 Wi-Fi readings, the access point data is calculated and checked with the internal database so that a location can be calculated. This process repeats itself after the location is calculated. So, if the accelerometer reading of position is off, the Wi-Fi localization system will set the position and the location will be corrected. It lowers the errors that can occur from just using accelerometer data.

f) Usage of the Accelerometer

When the user enters a building, the user can either select where they are in the building by touching the map, or they can let the Wi-Fi localization system show where they are. From there, the accelerometer is used to determine the position if the Wi-Fi localization system cannot pinpoint a location. The accelerometer works best when the back of the phone is level with the ground and the front is facing up. The acceleration on the Z axis is about the same as the acceleration due to gravity. The application then uses the X and Y coordinates to determine whether a significant movement was made in a certain direction and by how much.

First, the application needs to know what direction the user has moved. Since the application uses a grid structure to display the movement. The application does not need to know the exact position that a user has moved to. The application knows that the user is in a certain grid. It then needs to know which of the eight surrounding grids in the user has moved into.

Now, the application can tell by the sign of the acceleration in the X and Y directions in concert with data from the orientation sensor whether the person has moved northeast, southeast, northwest, or southwest. Then, the application looks at the acceleration in those directions. If it is close enough to 0, the application knows whether the person has moved north, south, east, or west. For example, if the user moves northeast according to the sign of X and Y, but not much in the Y direction, then the user moved more east than northeast.

Given that the application knows which of the 8 directions the user has moved in, the application just needs to know the distance that the user traveled. If the user travels over a certain distance in a certain direction, the application says that the user has moved into that grid space next to the current grid space in that direction. The application is essentially using a distance threshold to determine movement within a grid. Once the application knows that the user has moved to another grid, the process can start over again in order to determine the next grid space that the user moves to.

g) Usage of GPS

GPS is not used in the indoor navigation activity. The reason is that GPS cannot reasonably provide an exact location down to the grid where a user is located. Also, if a building has a lot of walls or ceilings that GPS must travel through, then the accuracy and the range of GPS are greatly reduced. This is not useful for indoor localization throughout the whole of a building.

h) Usage of the Orientation Sensor

The orientation sensor is used to detect what way the phone is facing. Using this information, it can mathematically calculate the acceleration in an X direction perpendicular to the north and a Y direction parallel to the north. This is very useful because no matter what direction the phone is turned, the application knows whether the phone is going away or toward the north direction and whether it is going toward the west or toward the east. It works in tandem with the accelerometer.

i) Usage of QR Codes

QR codes are used for localization indoors as well as getting information about a room. The user can scan the QR code of a certain class. The QR code contains a link to an XML file. This XML file contains the grid position of a room on an indoor map and details about the class that is going on and the start and end time. The application uses the position in the room to show the user where the room is located. When the user scans a QR code and the XML information is returned to the application, the indoor map shows the room that the floor is on and shows a person icon at that location. On the bottom half of the screen, it shows information about the room.

j) Usage of Wi-Fi

Wi-Fi is used by the application to determine the location of the user on the map without the help of GPS. The application calculates the minimums, maximums, medians, and means of the RSSI of all access points in range and compares them with those in the database for a specific building. It uses a cumulative distance measure. For example, if the database has a mean RSSI of 84.3 stored in the database for floor 2 of a certain building at position (5,4), and the application calculates a mean RSSI of 84.5, then the distance is 0.2. The distances are added up for each access point stored in the database at a certain position. The total distance is saved to a variable. So, if the total distance is 1.3 for position (5.4) and 10.6 for position (5.5), and 1.3 is the minimum total distance for all positions, then the application declares that your position is (5,4).

If there is not enough Wi-Fi information stored in the database, crowd sourcing can be used to gain the data. When a user is standing in a certain position, they can long press on the grid space where they are. The Wi-Fi data for their location will be stored in the external server. Then when another user stands in that same spot, the application will be able to determine their location and update the map.

Wi-Fi triangulation is not used. The reasons are mostly because of a lack of accuracy. Because of the various positions of the walls in the database, one part of a room may have three walls separating itself from the Wi-Fi access point, but another part of a room may have only two walls separating itself from the same Wi-Fi access point. A change in the number of reflections can alter the RSSI coming from a certain access point. Since the RSSI can vary from room to room apart from their distance to various access points, it is better to use a unique fingerprint that can determine your location.

V. Lesson Learned

A. Dual Screen emulator and Google APIs

At the beginning of the project we didn't have access to a dual screen device. We used the DTS Add-on for Kyocera Echo to launch an emulator of a double screen device. Unfortunately this emulator doesn't accept the Google APIs; the required library is not included in the emulator.Thus we had to begin the development of the outdoor navigation on a single screen emulator until we received a real device.

B. Overlays on the real dual screen device

The real dual screen device allows us to use the Google APIs 2.3.3 but one component didn't still worked: the itemized overlays. The log said that Eclipse used a different version of itemized overlay during the pre-verification. I tried all the Google APIs versions without success.

After several other attempts I found a solution. During all the previous attempts I used the DTS Add-on for dual screens as the base of the application and added the Google maps API as an external library. I tried to do the reverse: I used the Google APIs 2.3.3 as the base of the application and added the DTS Add-on as an external library and it works. Now the good version of the itemized overlays is used.

C. Problem to launch emulators on eclipse

When I started to program on the outdoor navigation, a serious problem occurs and prevents me to work on the project during 5 days. For an unknown reason all my emulators took almost 20 minutes to be launched and failed due to an error in the main android processes. I spend entire days to try to resolve this situation. I even uninstalled completely Eclipse and the SDK to restart from the beginning but the same problem occurred again. Finally I found d on a forum in internet that I just need to change a parameter in eclipse toallow more memory to the emulator. I don't understand why the problem occurred because I never changed this parameter before even during the first installation of eclipse.

D. Problem with the change of orientation on the dual screen phone

When I started to create the interface for the double screen phone I used a sample provided by Kyocera as a base but, after adding the Map View, the application crashed each time I tried to change the orientation of the phone because the application couldn't be killed. By reading again the documentation of the SDK of the dual screen I discovered that the sample I used avoided the destruction of the application it changed. I just disabled this precaution and all worked well.

E. Problem with GPS service

I didn’t receive data from the GPS service. My partner and I looked for the problem but we didn’t find it. Finally we decided to create a new thread in charge of the GPS and of the update of the current location.

F. Finding Unique Access Points in a Building

One of the parts of indoor navigation is finding your location using Wi-Fi data. This is sometimes difficult because Wi-Fi access points can have the same SSID but different MAC addresses. For example, Dragas Hall at Old Dominion University as of 2011, has over 5 access points with the same SSID. In order to find a unique access point the BSSID value is needed. This represents the MAC address of a certain Wi-Fi access point. The application just uses the MAC address because if the name of an access point changes, then a certain access point would not be found by the application when it may actually exist.

G. Variances in Accelerometer Readings

Accelerometer data is not just the acceleration of the phone relative to a certain position. It also takes into account the acceleration due to gravity. Also, this acceleration tends to oscillate very rapidly in a short amount of time. If a phone is tilted in the slightest bit, the acceleration due to gravity will change the accelerometer readings for the X, Y, and Z directions. It is difficult to eliminate the acceleration due to gravity because it is hard to measure the exact tilt of the phone and how gravity should be effecting each of the three accelerometer readings.

H. Using a Dual Screen Phone

I learned a lot about dual screen phones. There is a special SDK that must be downloaded in order to use it correctly. It has its own emulator, set of classes, and AVD. When using the Dual Screen class provided by the phone, the application could not be run on a single screen phone. The Dual Screen class could not be used. Also, if the SDK provided for the phone is not used, then there is no way to detect whether the phone has been closed and there is one screen or it is open and there are two screens. Also, getting Google Maps to work with it proved to be difficult. The maps.jar library had to be added as a dependency. Not all of the classes like ItemizedOverlay could be used.

VI. Future Work

A. Create More Floor Plans

It would be helpful to have access to more floor plans. Not all floor plans were publicly available. Some floor plans must be created from scratch. Future work entails using a laser measurer to record to sizes of rooms and use that information to create a floor plan for floors of certain buildings. There could be an activity on the phone where a user could add a floor plan for a building and have it work on the application.

B. Add More Information about Rooms and Classes

The application must have more and updated information about rooms and classes. The rooms and classes need to be updated when they change on the schedule. There needs to be a way to an external organization to edit rooms and classes quickly and easily. Each room needs to have information about it and it needs to be complete.

C. Use the Accelerometer to Gain Wi-Fi Data

Currently, the accelerometer is not able to update the Wi-Fi data based on the positions found by the accelerometer. In the a future release, it would be good to update the Wi-Fi access point RSSIs based on where the accelerometer determines that the phone is located. The accelerometer accuracy will have to be improved for this to be a good method for updating Wi-Fi data.

D. Add More Icons

Icons could be added at the bottom that shows the number of Wi-Fi signals that are found or something that shows the accuracy of the Wi-Fi based on the number of Wi-Fi signals that are found. Also, an icon could show whether GPS is available. Then GPS could be used to approximate the location of a user if Wi-Fi is not available.

E. Work on Both Single and Double Screen Phones

Since there are many more single screen phones on the market, it would be good if the application could work correctly on both single and double screen phones. There is both a single and double screen interface for the application. The single screen interface would have to be used for a single screen phone and then the references to the DualScreen library would have to be omitted.

F. Add Custom Places

It would be useful for a user to be able to add custom places to an indoor map. The places would not have to be for global viewing, but they could be linked to a unique ID for the phone, or to the name of the person. Then, when the user looks on the map, they will be able to find that location again. Also, they would be able to store custom places outdoors.

VII. References

A Star Algorithm For Path Planning
http://code.google.com/p/jianwikis/wiki/AStarAlgorithmForPathPlanning

ZXing: Multi-format 1D/2D barcode image processing library
http://code.google.com/p/zxing/

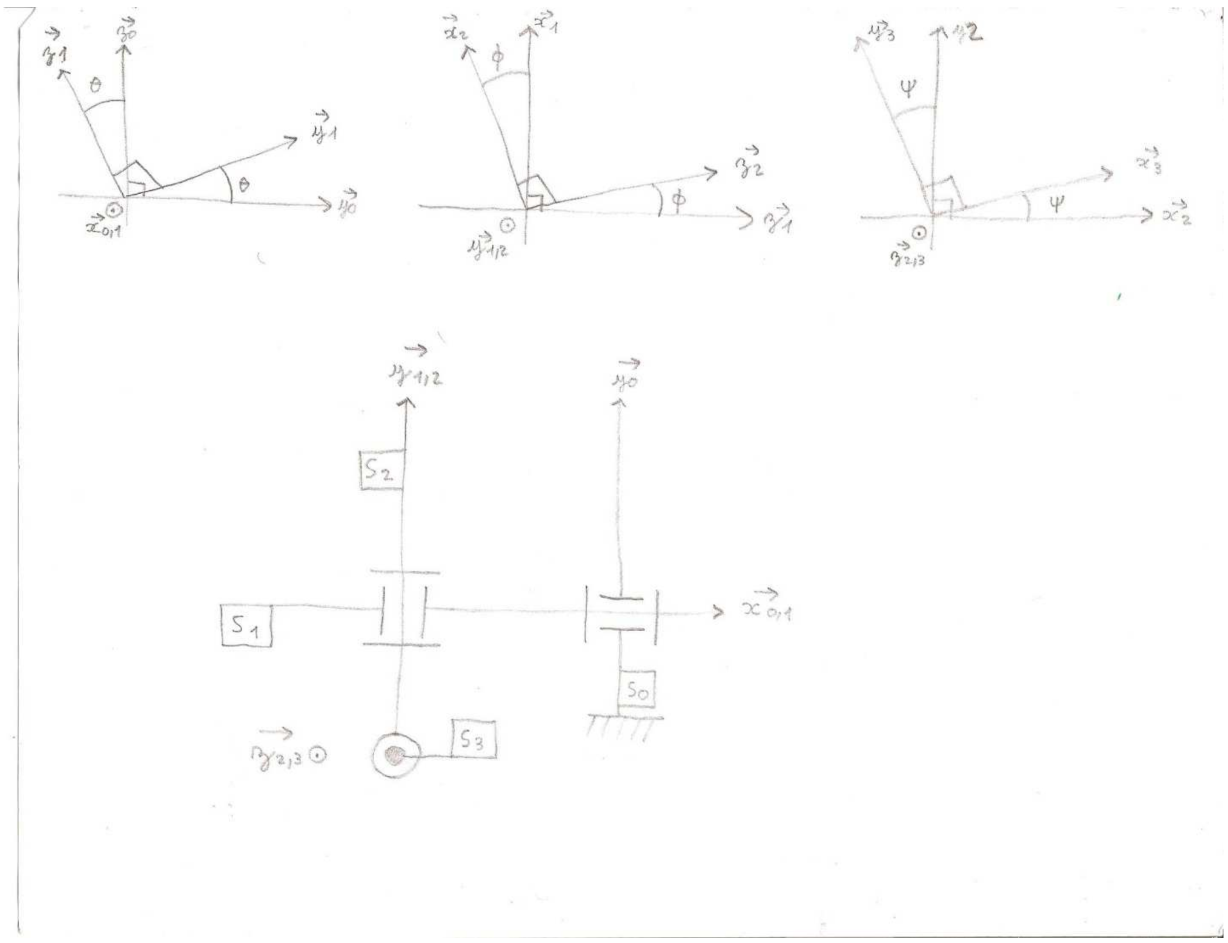
j2memaprouteprovider: a code for drawing a path on a map for outdoor navigation

http://code.google.com/p/j2memaprouteprovider/

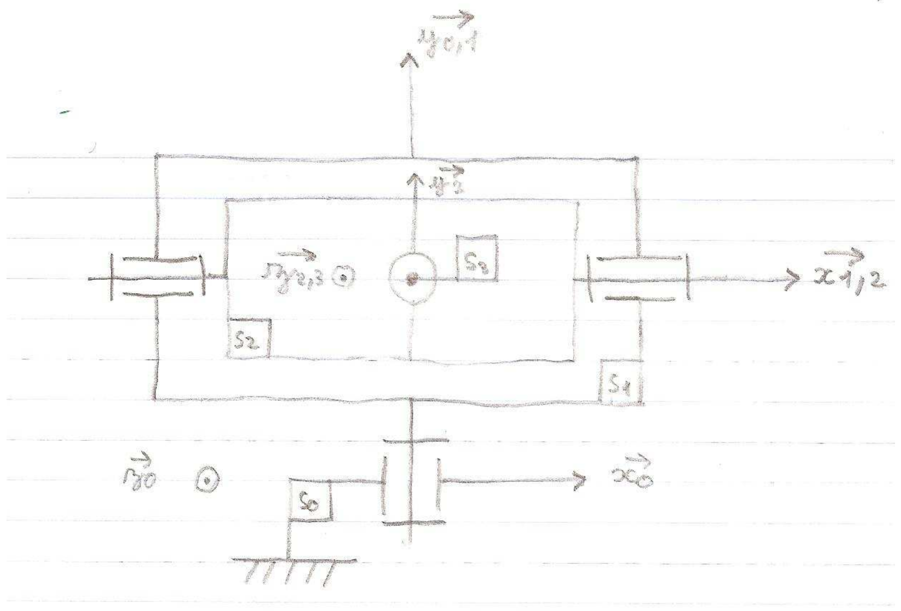
VIII. Appendix

A. Orientation simulator 3-Axis

1. Phone orientation model 1 and bases change figures



2. Alternative Phone orientation model (same results)



3. Bases changes 1/3

$1 \to 0$

$$\vec{x}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}_0 ;\ \vec{y}_1 = \begin{bmatrix} 0 \\ \cos\theta \\ \sin\theta \end{bmatrix}_0 ;\ \vec{z}_1 = \begin{bmatrix} 0 \\ -\sin\theta \\ \cos\theta \end{bmatrix}_0$$

$0 \to 1$

$$\vec{x}_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}_1 ;\ \vec{y}_0 = \begin{bmatrix} 0 \\ \cos\theta \\ -\sin\theta \end{bmatrix}_1 ;\ \vec{z}_0 = \begin{bmatrix} 0 \\ \sin\theta \\ \cos\theta \end{bmatrix}_1$$

$2 \to 1$

$$\vec{x}_2 = \begin{bmatrix} \cos\phi \\ 0 \\ -\sin\phi \end{bmatrix}_1 ;\ \vec{y}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}_1 ;\ \vec{z}_2 = \begin{bmatrix} \sin\phi \\ 0 \\ \cos\phi \end{bmatrix}_1$$

$1 \to 2$

$$\vec{x}_1 = \begin{bmatrix} \cos\phi \\ 0 \\ \sin\phi \end{bmatrix}_2 ;\ \vec{y}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}_2 ;\ \vec{z}_1 = \begin{bmatrix} -\sin\phi \\ 0 \\ \cos\phi \end{bmatrix}_2$$

$3 \to 2$

$$\vec{x}_3 = \begin{bmatrix} \cos\psi \\ \sin\psi \\ 0 \end{bmatrix}_2 ;\ \vec{y}_3 = \begin{bmatrix} -\sin\psi \\ \cos\psi \\ 0 \end{bmatrix}_2 ;\ \vec{z}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}_2$$

$2 \to 3$

$$\vec{x}_2 = \begin{bmatrix} \cos\psi \\ -\sin\psi \\ 0 \end{bmatrix}_3 ;\ \vec{y}_2 = \begin{bmatrix} \sin\psi \\ \cos\psi \\ 0 \end{bmatrix}_3 ;\ \vec{z}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}_3$$

4. Bases Changes 2/3

$\boxed{0 \rightarrow 2}$

$$\vec{x_0} = \begin{bmatrix} \cos\phi \\ 0 \\ \sin\phi \end{bmatrix}_2 ; \quad \vec{y_0} = \begin{bmatrix} 0 + 0 + \sin\theta\sin\phi \\ 0 + \cos\theta + 0 \\ 0 + 0 - \sin\theta\cos\phi \end{bmatrix}_2 ; \quad \vec{z_0} = \begin{bmatrix} 0 + 0 - \cos\theta\sin\phi \\ 0 + \sin\theta + 0 \\ 0 + 0 + \cos\theta\cos\phi \end{bmatrix}_2$$

$\boxed{0 \rightarrow 3}$

$$\vec{x_0} = \begin{bmatrix} \cos\phi\cos\psi + 0 + 0 \\ -\cos\phi\sin\psi + 0 + 0 \\ 0 + 0 + \sin\phi \end{bmatrix}_3$$

$$\vec{y_0} = \begin{bmatrix} \sin\theta\sin\phi\cos\psi + \cos\theta\sin\psi + 0 \\ -\sin\theta\sin\phi\sin\psi + \cos\theta\cos\psi + 0 \\ 0 + 0 - \sin\theta\cos\phi \end{bmatrix}_3$$

$$\vec{z_0} = \begin{bmatrix} -\cos\theta\sin\phi\cos\psi + \sin\theta\sin\psi + 0 \\ \cos\theta\sin\phi\sin\psi + \sin\theta\cos\psi + 0 \\ 0 + 0 + \cos\theta\cos\phi \end{bmatrix}_3$$

5. Bases changes 3/3

$\boxed{3 \rightarrow 1}$

$$x_3 = \begin{bmatrix} \cos\psi\cos\phi + 0 \\ 0 + \sin\psi \\ -\cos\psi\sin\phi + 0 \end{bmatrix}_1 ; \quad \vec{y_3} = \begin{bmatrix} -\sin\psi\cos\phi + 0 \\ 0 + \cos\psi \\ \sin\psi\sin\phi + 0 \end{bmatrix}_1 ; \quad \vec{z_3} = \begin{bmatrix} \sin\phi \\ 0 \\ \cos\phi \end{bmatrix}_1$$

$\boxed{3 \rightarrow 0}$

$$x_3 = \begin{bmatrix} \cos\psi\cos\phi + 0 + 0 \\ 0 + \sin\psi\cos\theta + \cos\psi\sin\phi\sin\theta \\ 0 + \sin\psi\sin\theta - \cos\psi\sin\phi\cos\theta \end{bmatrix}_0$$

$$y_3 = \begin{bmatrix} -\sin\psi\cos\phi + 0 + 0 \\ 0 + \cos\psi\cos\theta - \sin\psi\sin\phi\sin\theta \\ 0 + \cos\psi\sin\theta + \sin\psi\sin\phi\cos\theta \end{bmatrix}_0$$

$$z_3 = \begin{bmatrix} \sin\phi + 0 + 0 \\ 0 + 0 - \cos\phi\sin\theta \\ 0 + 0 + \cos\phi\cos\theta \end{bmatrix}_0$$