University of Waterloo

Software Engineering

**Design Decisions for Choosing Gestures**

Google

Mountain View, California

**Prepared by**

Noah Sugarman

Student ID: 20295111

User ID: nsugarma@uwaterloo.ca

4A Software Engineering

April 30, 2012

Noah Sugarman

23 East Terrace

South Burlington, VT

05403

Dr. Charlie Clark, Director

Software Engineering

University of Waterloo

Waterloo, ON, N2L 3G1

Dear Dr. Clark:

The enclosed report, titled *Design Decisions for Choosing Gestures*, is my fourth and final work term report. It corresponds with design decisions I made while employed for Google during my 4A work term.

Google is a large software company that deals with big data. I was a member of the accessibility team and worked on prototyping gestures. The set of gestures that met the design constraints and best fit the design criterion would be used to make an existing Google product accessible.

This report describes the process of analyzing the ease of performing a gesture and recommends the set of gestures should be implemented. Each gesture is evaluated for how well it satisfies the specified design constrains and criteria. The report’s audience is product managers employed at Google who are in charge of implementing the selected gestures.

I would like to thank my co-workers at Google for their advice and support during my time with them. I would especially like to thank Dominic Mazzoni and Alan Viverette for their excellent mentorship and recommendations.

I hereby confirm that I have received no help, other than what is mentioned above, in writing this report. I also confirm this report has not been previously submitted for academic credit at this or any other academic institution.

Sincerely,

Noah Sugarman

Student ID: 2029511

# Executive Summary

Google had an existing application called Application A that ran on mobile devices with touch screens. This application was used by millions of users, but was not accessible to blind users. To make the application accessible, gesture navigation had to be implemented. Gesture navigation would allow blind users to navigate around in the application by using gestures, rather then point and click. A successful gesture navigation implementation needed to support 14 unique gestures, be able to be implemented within 11 days and maximize the ease at which users could perform the gestures.

To solve this problem, several different gestures were prototyped and then analyzed using user testing. The time taken to implement the prototype gave an idea of the actual implementation time. The user tests provided an estimate of how easily the gestures were to perform. The focus of this report is on how the gesture evaluation data was used to choose a set of gestures to be implemented. Gesture implementation details and accuracy measurement algorithms are outside the scope of the report. The design criteria and constraints are discussed as well as why the accepted solution is favored over alternative solutions. The risk of implementing the accepted solution is also addressed.

The final solution was to implement four swipe gestures, eight L-swipe gestures and two pivot gestures. The basic details of how to perform these gestures are described in section 1.2*.* The gestures should be implemented in the order of swipe gestures, followed by L-swipe gestures and finally pivot gestures. Implementing the gestures in this order allowed for the bulk of the gestures to be implemented quickly. This maximized the time to test the gestures in a production environment. Following these recommendations ensured all design constraints were met and the gesture ease of use metric was maximized. Alternative solutions were advisable only if the design constraints changed.

Table of Contents

Executive Summary i

List of Figures v

List of Tables vi

1 Introduction 1

1.1 Purpose of the Report 1

1.2 Background Information 2

2 Problem Specifications 3

2.1 Detailed Problem Description 3

2.2 Design Constraints and Criteria 3

3 Evaluation 5

3.1 Method 5

3.2 Results 6

4 Solution 7

5 Conclusions 10

6 Recommendations 11

References 12

# List of Figures

Figure 1 - Gesutres implemented per day 9

# List of Tables

Table 1 - Evaluation of gesture groups 6

Table 2 - Top possible solutions 8

# 1 Introduction

## 1.1 Purpose of the Report

Google “is a global technology company focused on improving the ways people connect with information” [1]. Google had developed an existing application called Application A. This application ran on mobile devices with touch screens and was used by many users. The problem with this application was that it was unusable for blind users because it used point and click to navigate. To make the application accessible, a gesture navigation feature had to be implemented. This feature would allow blind users to navigate the application by using only gestures. The gesture navigation feature was required to support at least 14 unique gestures and be able to be implemented within 11 days.

Solutions that passed the design constraints were evaluated on one design criterion. The criterion was the ease of performing the gesture. The problem arose to determine which gestures should be implemented in order to meet the design constraints and evaluate highest against the design criterion. The problem of maximizing the ease of performing the gestures while remaining inside the two bounds from the design constraints made this problem an instance of a multidimensional knapsack problem.

This report covers design decisions made while researching an acceptable set of gestures to implement. The intended audience is product managers employed by Google who are in charge of the implementation of the gesture navigation feature. The report describes the method used to analyze the gestures, but, for confidentiality reasons, does not go into detail about how the gestures will be used by Application A. Design constraints and criterion that were used to evaluate possible solutions are listed, along with how well the possible solutions satisfied them.

## 1.2 Background Information

Accessibility is an important field in the world of software development. At Google, when an application is called accessible, it means it is usable by low vision users. This is usually done though navigation by voice commands, keyboard or gestures. Every command performed by a low vision user is paired with an audio clip that lets the user know they have performed an action and gives context on the current state of the application.

On touch screen enabled devices, navigation through gestures is the most common accessibility feature. There are a few common gestures used across many different applications at Google and other large companies. A swipe gesture is the most common. It involves the user moving their finger quickly across the screen in a straight line. An L-swipe or U-swipe is a swipe gesture in the shape of an L or U. A circle gesture involves the user tracing out a circle with their finger on the screen. A pivot gesture is when one finger remains stationary while the other finger moves in either a clockwise or counter-clockwise motion around it. Another type of swipe is the edge swipe, which is a normal swipe done close to the edge of the screen. Once the gesture detection is implemented, each gesture gets mapped to a navigation command. For example, swiping to the right might get mapped to the “select next” command. This removes the reliance on point and click navigation, therefor making the application accessible to low vision users.

# 2 Problem Specifications

## 2.1 Detailed Problem Description

Application A was an existing and working application that ran on touch screen based mobile devices and needed an alternative method of navigation in order to make it accessible to blind users. This alternative method of navigation was navigating with gestures. A deadline of 11 workdays was put on implementing the gesture. There were 14 navigation commands that needed to be implemented in order to make the application fully accessible, and so it was required to come up with 14 unique gestures.

28 gestures were prototyped. These 28 gestures could be split into six gesture groups. The gesture groups were thought of as the type of gesture. Each gesture in a group was identical except for the direction of the gesture. Gestures directions were kept at least 90 degrees apart from each other to increase accuracy and reduce performing a different gesture by accident. This meant that swipes could only be performed in one of four directions, and so the gestures in the swipe group were swipe north, swipe south, swipe east and swipe west. Because of the similarity between gestures in the same gesture group, implementing one gesture in a gesture group allowed for the remaining gestures to be implemented in negligible additional time. The six gesture groups were swipe, U-swipe, L-swipe, pivot, circle and edge swipe. The swipe and U-swipe gesture groups had four gestures per group, one for each cardinal direction. The edge swipe gesture group had eight gestures for swiping in one of two directions next to one of four edges. The L-swipe gesture group also has eight gestures for swiping in one of two directions after moving in one of four cardinal directions. Each gesture had to be analyzed to determine how easy it was to perform and how long it would take to implement. In order to find the best solution, the ease of performing the gestures had to be maximized while staying within the two design constraints.

## 2.2 Design Constraints and Criteria

Based on the behavior of Application A, the accepted solution had to provide at least 14 unique gestures. This minimum existed because Application A needed to provide 14 navigation commands in order to make the application accessible, and each of these commands needed to have a unique corresponding gesture. The second design constraint was that the accepted solution had to be implemented within 11 business days. This limit came from Google’s deadline on the deliverable.

A solution that passed the design constraints was evaluated based on one design criterion. This design criterion was the ease of performing a gesture, which was also called the ease of use. This was important because the easier the gestures were to perform, the easier it would be for blind users to use the application, which was the goal of the feature. Tie breaking design criteria were not considered because ties were impossible with how accurately the ease of performing the gestures was measured.

# 3 Evaluation

## 3.1 Method

To evaluate the different gestures, the ease of performing a gesture and implementation time was measured for each gesture. Implementation time was estimated by using the prototype implementation time and then adding half a day for optimizations. Implementation time was recorded for each gesture group, rather then for individual gestures. This was done because implementing additional gestures in a gesture group required negligible additional time.

The ease of performing a gesture was measured by performing user tests and keeping track of the accuracy at which users could perform the gestures. Twenty users took part in the user tests, and each user performed each gesture twenty times. Gesture accuracy was measured using a preexisting algorithm that determined what percent of the user’s gesture matched the desired gesture. Gesture accuracy was averaged over the twenty trails from the twenty users to determine a final ease of use score as a percentage. An ease of use score of 100% corresponded to all users perfectly performing the gesture in every trial. The gestures being prototyped were significantly different from each other, so there was no worry of confusing them with each other. This meant that gesture accuracy was a good metric for gesture ease of use.

Ideally, more users would partake in the user study. It was estimated that around 10,000 blind users would use Application A after gesture navigation was added. The ease of performing the gestures metric was measured using only twenty low vision users. This brought risk into the evaluation data because those twenty users may not accurate represent all users. With a population size of 10,000 and a desired confidence level of 95%, the number of users testing the application should have been 370 [3]. However, this was an unavoidable risk because Google did not have the resources to acquire 370 users to perform tests. With the sample size of 20, the confidence interval was 22%, which was still enough to give some idea of which gestures were better.

## 3.2 Results

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Number of Gestures** | **Ease of Performing the Gesture (accuracy)** | **Implementation Time (days)** |
| **Swipe** | 4 | 98% | 1 |
| **U-Swipe** | 4 | 88% | 2 |
| **L-Swipe** | 8 | 74% | 4 |
| **Pivot** | 2 | 95% | 6 |
| **Circle** | 2 | 73% | 4 |
| **Edge swipe** | 8 | 67% | 3 |

Table - Evaluation of gesture groups

Table 1 shows the results from the user testing and prototyping. The number of gestures column shows the max unique number of gestures that can be performed from each gesture group. For example, there are four gestures in the swipe gesture group because a swipe can only be performed in one of the four cardinal directions while keeping a 90-degree between gestures. User accuracy was the same across each gesture in a gesture group, so accuracy is displayed for the entire group rather then for the individual gestures.

From observing the data it becomes clear that swipe gestures were the easiest gesture to preform. This makes sense because it was also the simplest of the six types of gestures. Swipe gestures also took the least time to implement, which was also due to the gesture’s simplicity. The least easy to preform gestures were edge swipe gestures. This was due to blind users having trouble knowing where the edge of the screen was. Pivot gestures took the longest to implement. This was because the gesture involved multiple fingers and so a more complicated algorithm was required. However, pivot gestures had the trade off of being extremely easy to preform. Because swipe gestures were the easiest to preform and took the least time to implement, they were very likely to be included in the final proposed set of gestures.

# 4 Solution

The ideal solution was a combination of gestures that had the best overall ease of use and meet all design constraints. This problem was equivalent to the multidimensional knapsack problem. An object’s “weight” was the ease of performing the gestures and the two bounds were the constraints on implementation time and required number of gestures. This problem was classified as an NP-hard optimization problem [2]. This instance of the problem had a small number of possible solutions, so it was easy to solve with a brute force method.

The brute force method was performed by first forming solutions from every possible combination of gestures. Solutions that did not meat the design constraints were removed. All gestures had positive ease of use metrics, which meant the optimal solution was a combination such that no other gesture could be added without going over one the constraints. The sets of gestures that failed this test of optimality were also removed. Then the set of gestures with the least implementation time was added to the list of possible solutions. This left the three solutions shown in table 2. The “X” in figure 2, located below the name of a gesture group, denoted the corresponding gesture group was part of the solution. The accepted solution was solution two because it had the highest ease of use, up 1% from the second best solution.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Sub-algorithms** | | | | | | | |
|  | **Swipe** | **U-Swipe** | **L-Swipe** | **Pivot** | | **Circle** | | **Edge Swipe** |
| **Solution 1** | 4 | 4 | 8 | 0 | | 0 | | 0 |
| **Solution 2** | 4 | 0 | 8 | 2 | | 0 | | 0 |
| **Solution 3** | 4 | 4 | 0 | 0 | | 0 | | 8 |
|  |  |  |  |  | |  | |  |
|  | **Number of Gestures** | **Overall Accuracy** | **Implementation Time (days)** |  |  | |
|  |  |  | |
| **Solution 1** | 16 | 85% | 7 |  |  | |
| **Solution 2** | 14 | 86% | 11 |  |  | |
| **Solution 3** | 16 | 82% | 6 |  |  | |

Table - Top possible solutions

Calculating the number of gestures per gesture group divided by implementation time gave a gestures implemented per day metric. This was used to determine the order in which the gestures were implemented. Figure 1 shows a graph of this metric. From this graph it is clear that implementing swipe gestures gives the most gestures implemented per time spent, followed by L-swipe and finally pivot.

Figure – Gestures implemented per day

# 5 Conclusions

The accepted solution was to implement all possible swipe gestures, L-swipe gestures and pivot gestures. The implementation time was 11 days and the average accuracy at which users could perform the gestures was 86%. This solution was the best because the gestures were the easiest for users to perform while staying within the design constraints. Solution 3 had better implementation time, but much worse ease of use. Based on the gestures implemented over time metric, swipe gestures should be implemented first, followed by L-swipe gestures and finally pivot gestures.

Following the accepted solution completely solves the problem, however there were some risks. The ease of performing the gestures was measured based on data from only twenty users. This small sample may not accurately represent the entire user base of Application A. This caused the risk of the evaluation data not properly representing the average use case. Another risk was if something goes very wrong in the implementation, there was no extra implementation time to spare because the expected implementation time equals the max implementation time allowed. Despite these risks, the data should still give a good idea about how the solution will perform.

# 6 Recommendations

It is recommended that the Google Application A team implement swipe gestures, L-swipe gestures and pivot gestures. If a different set of gestures is implemented then the gestures will be harder to use or the solution will not meet the design constraints. The team should implement the gestures in the order descried in this report. Doing this ensures the most number gestures are implemented in the least amount of time. The accepted solution should be finished within eleven days, which leaves no extra time to spare.

If the design constraints change then an alternate solution may be optimal. For example, if the constraint of needing 14 gestures was increased to 16, then the optimal solution would be solution one. On the other hand, if a small implementation time is critical, then solution three may be preferred as it has the lowest implementation time. Solution one and solution two only differ in ease of use by 1%, so if there are any concerns about implementation time increasing or number of gestures needed increasing, then solution one should be taken.

# References

[1] Reuters. “Google Inc (GOOG.O).” Internet: <http://www.reuters.com/finance/stocks/companyProfile?symbol=GOOG.O> [May 3, 2012].

[2] P. Chu and J. Beasley. “A Genetic Algorithm for the Multidimensional Knapsack Problem”. *Journal of Heuristics*, vol. 4, pp. 63-86, 1998.

[3] Creative Research Systems. “Sample Size Calculator.” Internet: <http://www.surveysystem.com/sscalc.htm> [May 3, 2012].

Acknowledgements

I would like to acknowledge Alan Viverette for helping me set up the user studies used in the gesture evaluation. I acknowledge Google for letting me use their computers, software and office space for my work. I would also like to acknowledge the staff of the Software Engineering department and the staff of CECS at the University of Waterloo for their work report guidelines.