Scaling Android User Interfaces

A Case Study of Squid

David S. Janzen
California Polytechnic State University
San Luis Obispo, CA USA
djanzen@calpoly.edu

Andrew Hughes Anthony Lenz
Steadfast Innovation, LLC
San Luis Obispo, CA USA
{andrew,tony}@steadfastinnovation.com

Abstract

Modern mobile device screen sizes vary significantly, from watches to phones to tablets. Android in particular has supported varying screen sizes since 2009, and is currently being installed on large format touch displays over eight feet diagonally. These extreme variations present significant challenges to app developers who desire to support all devices within a single application. Squid is a handwriting, note-taking application with well over two million installations and a 4.2+ star rating in Google Play. Although originally designed for tablets, we have adapted Squid to run effectively on phones and more recently large format displays. In this paper we report on UI/UX challenges that we faced and decisions made to overcome them.

Categories and Subject Descriptors H.5.2 [*User Interfaces*]; D.2.2 [*Design Tools and Techniques*]

Keywords mobile computing, software engineering

1. Introduction

Android has supported varying screen sizes since version 1.6 (API Level 4, Donut)[1] released in 2009. At the time, phone screen sizes varied only slightly. However, with the introduction of tablets, watches, and large format touch displays, screen sizes can now vary from around 1 inch to over 100 inches. Squid[2], originally named Papyrus, is an example of a popular Android app that is released in a single binary that runs on devices from phones to large format touch displays. Squid was first released in 2012 and is developed by Steadfast Innovation, LLC. The authors are the co-founders and lead software developers of Squid.

2. Squid Overview

Squid is a handwriting note-taking app designed to retain the natural feel of pen and paper while adding features that leverage modern digital device capabilities. Squid allows users to create fixed or infinitely-sized multi-page notes on over sixty provided backgrounds and user provided PDFs. Tools include highly customizable pens, erasers, shapes, text, and a selection tool. Figure 1 demonstrates the color picker tool above a PDF that is being annotated. Strokes are stored as vector graphics so they can be individually selected and edited, and a high quality rendering can be achieved at any zoom level. Images can be imported and cropped. Notes can be exported as PDFs or images, and notes can be saved locally and backed up to cloud providers. Notes can also be cast to media devices such as televisions and projectors.

Uses of Squid often vary by device size. Phone users typically create to-do and shopping lists, draw maps or doodles, and sign documents. Tablet users are often students or business professionals who record lecture or meeting notes, give presentations, and markup PDFs for grading or editing. On large format touch displays, Squid is primarily used as a digital whiteboard in classrooms and conference rooms.

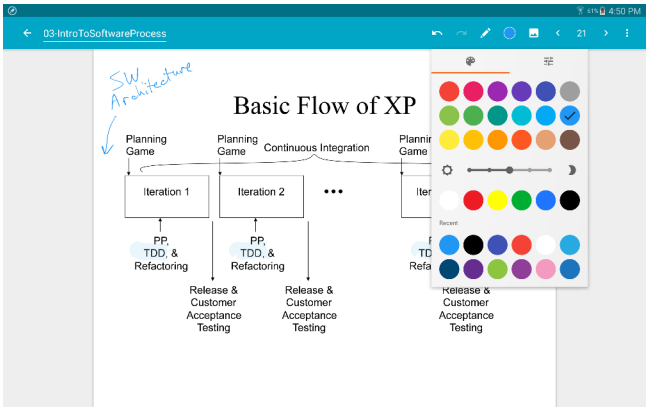

Figure 1. Tablet Horizontal Note View.

[1] `https://developer.android.com/guide/practices/screens_support.html`

[2] `http://squidnotes.com/`

Mobile!'16, October 31, 2016, Amsterdam, Netherlands
ACM. 978-1-4503-4643-6/16/10...
http://dx.doi.org/10.1145/3001854.3001859

3. Scaling Down to Phones

Squid originally targeted tablet devices with active pens. When the Samsung Note was released in 2012, Squid was adapted to work on phones as well. The smaller screen introduced a variety of challenges. For example, the note browser on the phone interface is limited to two columns, while the tablet accommodates three columns in a vertical orientation. The differences are exacerbated by the fact that a vertical orientation is more commonly used with phones, and a horizontal orientation is more common with tablets. Figures 2 and 3 demonstrate these differences between the two column vertical phone page management interface with the seven column horizontal tablet version. For large documents, the tablet version makes selecting, deleting, and rearranging note pages much simpler. Our approach uses a custom autofit RecyclerView that calculates the number of columns based on thumbnail sizes that fit within minimum and maximum widths based on screen sizes.

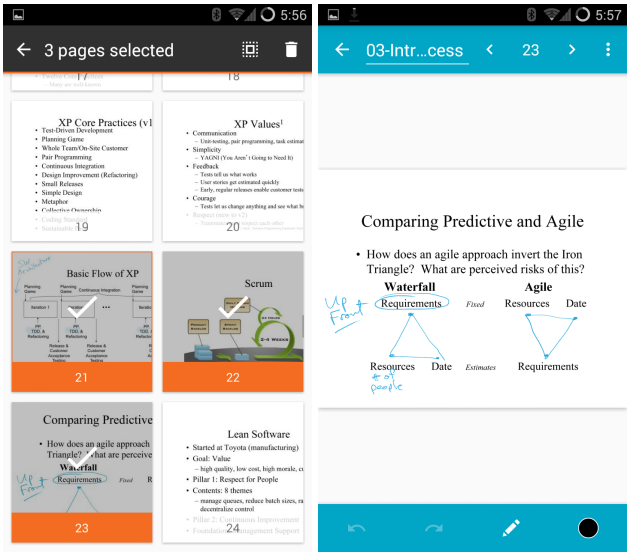

Figure 2. Vertical Phone Note Editing and Page Management.

The vertical phone orientation imposes severe limitations on what can be included on the action bar. By default, Android moves items that don't fit on the action bar into the overflow menu. We elected instead to create a second action bar on the bottom of the screen for small devices. Figures 1 and 2 compare the note editing UI on the tablet that has ample room for all the tool icons with that of the phone that is split between the top and bottom action bars.

4. Scaling Up to Large Interactive Displays

Squid is now being pre-installed on many large format touch displays which are becoming increasingly popular. The quality of the inking was the first challenge with large displays. The default Squid pen tool produced natural-looking variable-width strokes on smaller displays (e.g. 10 inch

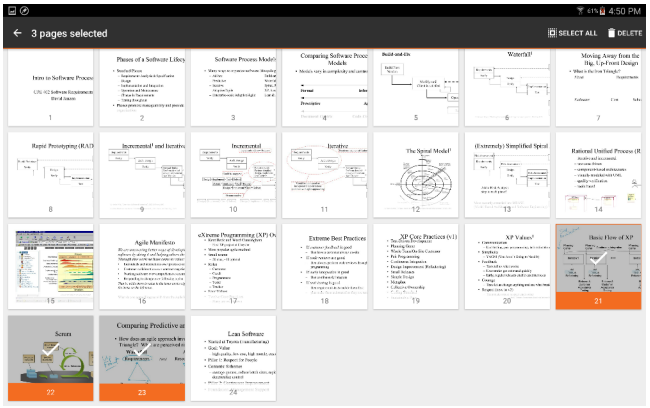

Figure 3. Horizontal Tablet Page Management.

tablets) with active pens. This was achieved with vector graphic strokes, a point reduction algorithm, and using the active pen pressure sensitivity to adjust the width of strokes. However, with the large size and only capacitive pens, individual segments in strokes were visible and strokes lacked the natural width variation.

To overcome this challenge, we created a new velocity-based pen tool that adjusts the width of the stroke based on the speed of the writing. Although the concept is not new[3], its necessity in Squid was prompted by the scale of the UI.

Due to previous efforts to support multiple screen sizes on the small scale (tablets vs phones), no additional UI changes were needed in Squid to work properly on very large displays. We had already made use of Android facilities such as layout aliases, density independent pixels, and nine-patch bitmaps to support multiple screen sizes.[4] We did integrate a new physical pen with shortcut buttons for large displays that helps to minimize travel time to the tool picker.

A few challenges remain with scaling to large displays. For instance, multi-touch on small displays is typically restricted to a few fingers, enabling gestures like pinch to zoom. However, large display OEMs advertise that they can recognize ten, twenty, and even forty or more touch points. Supporting multiple simultaneous users presents interesting questions. For example, what should the app do when a tool is changed (e.g. pen color change)? Should the change apply to all users? Or, how should pan/zoom be applied when multiple users are writing? Should it be disallowed, or does pan/zoom preempt other writers, or something else? Should users be able to select multiple tools for simultaneous use (e.g. one person using a pen while another using an eraser)? Qeexo's TouchTools[5] is one attempt at allowing multiple tool usage based on distinct gestures.

[3] http://www.graficaobscura.com/dyna/

[4] https://developer.android.com/training/multiscreen/

[5] http://www.qeexo.com/press/2016/qeexo-launches-touchtools-virtual-tools-for-smart-devices