Comparing Menu Designs

Report and Documentation

Henry Dixon
hdixon@cmu.edu

Spring 2016

Prepared for
Dr. Brad A Myers
05-440 *Interaction Techniques*

1. Magic Lantern

1.1 Background

Magic Lantern is a third-party camera firmware for Canon's line of DSLR cameras that can be likened to "jailbreaking" the cameras to allow for a few extra features and more fine control over the camera's operation. Magic Lantern is an open source, unofficial project which can be downloaded from Magic Lantern's website (magiclantern.fm). I used Magic Lantern on my Canon T2i.

1.2 Properties

After you turn on your camera, you can press an icon labeled with a trash can to enter Magic Lantern's menu system. The menu immediately visible is a vertical linear menu; users can select options by pressing the up and down arrow keys adjacent to the screen. However, one can move between different vertical linear menus, each dedicated to a certain set of functions (e.g. "Audio," "Exposure Control," etc.). This movement is accomplished by pressing the right and left arrow keys.

1.3 Advantages and Disadvantages

Because of its nature, Magic Lantern lives in a niche of Canon power users — people willing to put in effort to download and install a third party camera for their camera. As such, the usability of the product is underemphasized for the focus on increased functionality. Additionally, the whole system can

feel odd because it works on top of the default menu system, which leaves Magic Lantern's designers to simultaneously try and adhere to the mental model established by the default menu system without using operations reserved for the default one. As an example, if a user wants to evoke the Magic Lantern menu, she needs to press the button usually used to delete photos, denoted with a trash can. These unintuitive actions need to be specifically learned (rather than intuited), which usually indicates poor design. In this case, Magic Lantern's target audience is dedicated to gaining granular control over their camera in a way that the average user would not be motivated to do. These users are willing to accept Magic Lantern's learning curve, which affords professionals control in spite of the system's less-than-ideal design.

2. VSCO

2.1 Background

VSCO is an app freely available for iOS and Android, designed for photographers who are trying to edit their smartphone photos on their smartphone. The app can be downloaded from the iTunes Store (App Store). In this paper, I will be specifically focusing on the process of editing an individual photo once it is selected from the library, as this is the primary function of the app.

2.2 Properties

When editing a photo, VSCO presents a few different options and modes for editing, depending on the kind of edit you want to make. Often, users want to simply add a filter and/or adjust particular attributes of the photo (e.g. exposure, contrast, tint, white balance, crop, rotation, etc.). When the user has a photo selected, she is faced with four buttons: an x (which escapes from viewing an individual picture, reverting to the entire library), two tiny sliders (which begins the editing process), an arrow (which has a few other functions without an obvious grouping), and a circle of circles (which affords a few different functions, also without an obvious grouping). When you tap on the two sliders to begin editing, VSCO replaces the buttons with filters, represented by thumbnails of the photograph with specific photo edits applied for each filter. To specify a filter, the user can tap on a filter to preview it on the picture. The user also has an option to swipe right and left across the filters, revealing different filters unable to be presented immediately. To set the magnitude of the filter's effect (i.e. how intense the filter is), users can tap on the filter thumbnail twice to reveal a slider to adjust the filter intensity. To make non-filter-related edits, you need to find and tap the tiny white arrow below the filter thumbnails,

apparently indicating to the user that there is something able to expand from the bottom of the screen. This reveals a tray of four more icons, as well as a check and an "X", allowing the user to save or discard changes at any time. Initially, the paint brush icon is selected to indicate that we're in filter editing mode. To edit other attributes of the photo, the user must tap the wrench icon. Upon tapping the wrench, the user is greeted with a set of icons which behave the similarly to the filter selection screen. We have the option to swipe along the line of icons to reveal more options, and we can tap on any icon to edit the corresponding attribute. Nearly all of the attributes able to be edited (e.g. exposure and contrast) conform to VSCO's slider-everywhere paradigm. The sliders even permeate operations like rotation, which is sometimes accomplished using a more tactile interaction (e.g. using two fingers to rotate the photo as though it were a piece of paper on a table).

2.3 Advantages and Disadvantages

The VSCO menu flow is consistent, perhaps to a fault. The menus have a lot of depth stemming from the high-level edit selection buttons. On each level in the hierarchy, the user is consistently presented with either four tappable buttons or a sliding row of tappable options, most revealing a slider to tune the effect. The consistency in the menus make it easy to conceptualize, despite the sometimes-enigmatic icons. The downside of all this consistency is that some operations which may be better accomplished with a different interaction are all performed the same way, with a slider.

3. Apple Pay

3.1 Background

Apple Pay is a contactless payment and wallet service by Apple which allows users to make payments by tapping their phone to a payment terminal. The service is included in all iPhones beginning with the iPhone 5. The service is also available for the Apple Watch, but today I will be looking at the iPhone 6 implementation of the service.

3.2 Properties

To access Apple Pay, a user can tap the home button on their iPhone twice when it is locked. To complete a transaction, the user sees a prompt: "pay with touchID," accompanied by the image of a fingerprint and the card to be

charged. At the bottom of the screen are any other non-default cards; when tapped, these cards expand from their compressed stack and afford the user an area to tap and select a given card. The user is expected to rest her finger on the home button/touchID fingerprint reader; after the system detects the user trying to use touchID, the prompt is replaced by one telling the user to move the phone towards the payment terminal to complete the transaction. If the authentication is unsuccessful with touchID, the phone vibrates and the whole menu shakes, telling you to try again and showing the fingerprint icon. After three failed attempts, the user is told to simply enter her passcode.

3.3 Advantages and Disadvantages

When Apple Pay works in the real world, the process feels magical. However, the the menu structure is abstract and not self-explanatory, and many operations are completed by less-familiar interactions like resting a finger on the touchID sensor and physically moving the phone into proximity of the payment terminal. These interactions feel fluid and intuitive, yet not totally comfortable or reliable. The best example is how often a user can accidentally access Apple Pay; when trying to unlock your phone, you have to press the home button. Sometimes, the home button does not register immediately so you have to press it again and end up with the Apple Pay. Another abstract concept from the Apple Pay interface is the operation of paying with a card that is not the default. Personally, it took me a very long time to figure out that I was supposed to tap on the small stack of cards at the bottom to expand the stack and select a new card.

4. Spark

4.1 Background

Spark is a mobile email application that emerged after the recent surge in that area (lead by Dropbox's Mailbox). Anyone with an iPhone can download the application via the Apple App Store.

4.2 Properties

Spark takes many tips from the currently popular email-client design brought forward by Mailbox and now popularized by Google's material design take on email, Google Inbox. The concept of the app is to achieve so-called "inbox zero," where you have either archived, deleted, or temporarily hidden your emails until a specified time, leaving the user with an empty inbox. The flow of the app and

its menus reflect this purpose. When each email comes in, the user is presented with a traditional-enough vertical menu listing all of the emails in one's inbox. Unlike most such menus in other contexts, Spark allows the user to swipe right or left on each email. Swiping right immediately archives the email. If you need to take action on an email, but not immediately, swiping left offers another vertical list menu of options with actions a user can take. Like nearly all email clients, you can tap on an email to view and respond to it.

4.3 Advantages and Disadvantages

Spark's menu design is honest, efficient, and conducive to its goal as an application. The system excels in its simplicity, both in concept and execution. There are very few actions the user can take on an email (by design). Swiping is a quick action you can take to efficiently work through all of your emails without wasting time. Moreover, the actions you can take are echoed in many places in the app, underscoring the conceptual clarity of each operation. Spark has few downsides as far as menu design: it's consistent, honest, and elegant.

5. Snapchat

5.1 Background

Snapchat is a wildly popular application that can be downloaded from the Apple App Store on a variety of devices. As with the rest of the iPhone apps, I tested Snapchat on my iPhone 6.

5.2 Properties

Over the years, Snapchat has really refined the experience to be conceptually simple while offering a range of functionality. When opening the app, the user is greeted with a screen showing the camera, and the design emphasizes the large shutter button, which allows users to take a picture to send to others: the core of Snapchat. The "menus" that make up Snapchat do not resemble traditional menus, and most of the work done by the user is by swiping. The app is laid out such that from the initial camera screen, the user can make one swipe to get to other Snapchat environments: swiping right allows you to view your messages, swiping up allows you to do things related to adding and managing your friends list, and swiping right allows you to view Stories (another one of Snapchat's core features). When viewing stories, you can swipe right once again to reach the "Discover" page. This page has not been integral to Snapchat in the past

(and many would argue that it's not a core feature), so it makes sense that it takes two swipes to access.

5.3 Advantages and Disadvantages

The genius in Snapchat's menu design comes from the conceptual layout of the application coupled with how quickly a user can move from one core feature to another (with just a swipe or two). Once a user becomes acclimated to how the application is organized, she can quickly and effortlessly navigate the multiple modes. The disadvantage to such a abstract (yet simple) design is that at first, the user might not be able to make a mental model and remember exactly where each screen is located in relation to the others. However, this does not prove to be a large concern due to how quickly someone can search through all the screens. Moreover, Snapchat is the kind of app that people use nearly every day, so any learning of the menu setup will be accomplished quickly.