Cover

Hello! My name is Ricardo, my Group colleague is Rafael, and we are going to present you the Paper UI Design Patterns. Despite of this Paper being written in 2009, it still addresses lots of nowadays problems concerning Mobile UIs as you will see! We hope you enjoy!

Motivation

- The Design Patterns we are going to present you, are really useful for developers that are starting to develop Mobile UIs, which may be the case for many of you. The reason they are so important, especially in this phase, is because they can offer you a clear way of solving well known issues that you are going to start to face.
- Following these Design Patterns might increase the usability and adoption of your Application, solving the problems you are faced with, with Standard solutions, which can save you trouble, and therefore speeding up the UI development.
- For a given problem there are lots of possible Design Patterns, which one is better for you to pick may depend on the Application you are building. And whenever you have a complex problem, you can apply our "Divide to Conquer" rule, originating a set of more specific problems for which you can combine these Patterns all together.

Introduction

- We are going to present you 6 well-known problems out of 26 associated with these Areas. These are problems that you are likely to start to face when building Mobile Applications. For which one, we are going to explain a few Design Patterns that could be adopted by you to solve its issue.
- The problems are grouped into 3 main Problem Areas.
    - The 1st Area is related with the problem of managing the Screen Space of the Mobile Device, many of you can be used to develop UIs for Computer Applications, where the management of the Screen Space might not be a significant problem, but when it comes to building Applications for Palm-Sized screens or less, things change.
    - The 2nd Area is related with the problem of managing the User Interaction, nowadays Androids have no HW Keyboard, and whenever a User needs to enter text, a SW Keyboard pops into the screen, further restricting the screen space available. This Area also addresses the problem of making UI controls sufficiently large so that the User can interact with it, with its finger, even in a restricted screen space.
    - The 3rd Area is related with the problem Designing at Large where we explore how to properly brand an UI Application, and how to deal with Long-Lasting Operations, in order to display proper Feedback progress.
- The 1st part of our presentation will be exploring with you, these 6 individual problems, which are from different Areas, covering therefore specific problems of all

these main Areas, giving a big picture of each one. For each problem we explore its general guidelines, as well as some important Design Patterns.
- Then we are going to present you the results of a questionnaire filled by developers with mixed Background, about the usefulness and relevance of these Patterns Collection.

Interacting With Applications Without Using Stylus

- First of all, Stylus is a Special Kind of a Pen that is meant to help the Marking and Writing on Touch Devices.
- However, when building Mobile Applications, it is important to bear in mind that there are Users that do not use Stylus, because they simply prefer not to use it, nowadays this is the most common, or there may be cases where it is unfeasible, for example, when they only have one hand available.
- The problem is that when the User uses his finger instead of a Stylus, it conceals larger parts of the UI, worsening with the possibility of wearing gloves, consequently its accuracy of Pointing is coarser, making more difficult for the User to hit small UI controls.
- The most obvious solution is to make UI controls bigger. However, it can cause drawbacks and be unfeasible in certain cases. There is always a Trade-off of Cost VS Benefits.
- When building an Application, the developer could simply choose most appropriate UI controls, such as:
  - Use bigger Buttons.
  - Triggering Checkboxes and Radio Buttons when the User clicks in any part of the control, instead of listening only to clicks on the Tick Box.
  - Increase the items size of List or Combo Boxes. However, it may raise the need of using Scrollbars.
- In certain cases, adapting the UI may not be that simple, advanced adaptations can be done by making Subclasses of existing UI controls, which is usually easier than make self UI controls from scratch. The difficulty of doing it depends according to the Platform and tools available.

Interacting With Applications Without Using Stylus
Design Pattern: Finger Friendly Menu Choices

- There are Finger Friendly Design Patterns for any kind of UI controls, the one I am going to talk is an example of it, the Finger Friendly Menu Choices.
- The solution to make Users able to operate an Application using their finger is, instead of making Menus always visible, provide Menu Choices in a small Popup as illustrated in this Figure, which allows for the Buttons to be bigger than they could be, if they were always visible, because it would occupy a larger portion of the screen space.

Design That Supports Branding, Is Aesthetic, And Utilize Screen Space Optimally

- There are Organizations that want its Applications to apply their own branding to the Products. There are some challenges about Branding of UIs because it may be difficult to keep a Standard look and feel, consequently it may cause usability issues, making the UI more difficult to learn by the majority of Users. In the case the branding either adds Visual Elements or customizes UI controls it can become a screen space issue and decrease its affordance.
- There are several branding mechanisms, such as displaying the Company logo in the UI, fill the background with the Company colours, or a more subtle approach could be using a Company specific UI design that is recognizable by its Users.

Design That Supports Branding, Is Aesthetic, And Utilize Screen Space Optimally
Design Pattern: Brand The Standard

- This Pattern is particularly useful when not just branding but also compliance to the Standards is desirable. It can be done by adding or customizing branding elements to the Platform Standards instead of building ones from scratch. The brand should be subtle so that the User is not overcrowded.
- However, it can make it difficult to recognize.
- Examples of such approaches are:
    - Changing background colours.
    - Adding a Pattern whose example can be seen in the Figure, where the same Colours Pattern of the logo appear on the Application buttons.
    - Or using a specific Font which is characteristic of a brand.

Design That Supports Branding, Is Aesthetic, And Utilize Screen Space Optimally
Design Pattern: Branding The Controls

- This Pattern is usually applied when branding is more important than following the Standards.
- It has the advantage of potentially occupy less screen space than the previous approach, because custom UI controls are developed specially for a given Application.
- However, the main drawback of this approach is the cost of implement such specialized UI controls potentially from scratch, being more expensive than the previous approach.

User Interaction During Waiting For Long-Lasting Operations To Complete

- It is important to ensure Good Feedback solutions for Long-Lasting Operations on Mobile Applications, because usually a User is less patient on its Mobile Device than on its PC, and Good Feedbacks make the User more patient.
- However, most of the times there is no sufficient information on the Mobile Device which allows to display an accurate progress indication. Aggravating this situation,

due to Mobile Devices are usually connected through Wireless Networks, there is a sharp bandwidth oscillation. Also, as its computational power might not be so powerful as on a PC, the necessary computation for calculating an accurate progress indication, can be a significant overhead on such devices.

- There is usually a Trade-off about whenever displaying a better feedback solution is preferable over better performance.
- That said there are no optimal solutions. A possible approach could be avoiding or bypassing those Long-Lasting Operations. To do so there can be combined both data caching and pre-fetching that could be performed under the hoods, before the User asks for it, and continue its transfer while the User is interacting with it. Even if it reduces the Application performance it may be preferable over making the User wait.
- However, there can be cases when avoiding or bypassing such operations might not be possible, in that case there can be followed three approaches:
    - The 1st, is to display to the User a Wait Cursor with a User-Friendly message, for example, "Please Wait (…)". Which can be seen at the Figure at the Left.
    - The 2nd, is to inform the User that some operation is being performed, displaying a progress indication which can be achieved by a counter or a slider that shows the time spent percentage. Which can be seen at the Figure at the Right.
    - The 3rd, is the approach presented in the following Design Pattern.

User Interaction During Waiting For Long-Lasting Operations To Complete
Design Pattern: Inform The User About What Is Happening

- This Pattern is useful either when it is really necessary to inform the User of what is happening under the hoods, or when the information to do so can be easily obtained.
- This may be implemented by a scrolling text that the User can scroll through to see the oldest relevant events, and the new ones are appended at the top which the User can see without the need of scrolling. The second way to do so is to display a single text that is being constantly overwritten with new relevant events at the time they happen. Which can be seen at the Figure at the Right.
- Regardless of what approach is used, the information that is displayed should be comprehensible by all Users, not much technical.