ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Department of Informatics

Human-Computer Interaction

Team:

- https://github.com/evvalvis
- https://github.com/htsili
- https://github.com/arisfkiaras

Source code: http://bit.ly/1Lc1fl6

Video: https://www.youtube.com/watch?v=x9ibwT3b450

Contents

1. Project Understanding

1.1 The target audience of the application

As a group of young aspiring developers with an aptitude in the design of user interfaces, we always seek for ways to improve the user experience of our users. In order to further expand our knowledge about the designing of user interfaces we take the course of Human-Computer Interaction. During the course we chose to design a mobile application aimed at kids from the ages 6 to 8 years old. These days, kids are on computers almost as soon as they can sit up and move a mouse or tap a screen. It's now common for a 7-year-old kid to be a seasoned Internet user with several years of experience. We thought it would be an interesting project, as we needed to consider the differences between how a child uses a mobile compared to an adult or even a teenager. Finally we chose to develop a launcher and not just a single app because nowadays the navigation inside a mobile operating system especially in Android is both complex and frustrating for young kids.

1.2 User Needs Analysis

After deciding the type of application we wanted to develop, we proceed into analyzing the needs of our users. The first challenge we faced was the need to target very narrow age groups when designing for children. There's no such thing as "designing for children," defined as everybody aged 3–12. At a minimum, you must distinguish between young (3–5), mid-range (6–8), and older (9–12) children. Each group has different behaviors, and the users get substantially more web-savvy as they get older. And, those different needs range far beyond the obvious imperative to design differently for pre-readers, beginning readers, and moderately skilled readers. We found that young users reacted negatively to content designed for kids that were even one school grade below or above their own level. Children are acutely aware of age differences: at one app, a 6-year-old said, *"This app is for babies, maybe 4 or 5 years old. You can tell because of the cartoons and trains."* (Although we might view both 5- and 6-year olds as "little kids," in the mind of a 6-year-old, the difference between them is vast).

In order to better understand the needs of our target audience we had to make some experiments to observe the main differences in user behavior between children and adults. The following table summarizes the results.

	Children	Adults
Goal in using apps	Entertainment	Getting things done Communication/community
First reactions	Quick to judge app (and to leave if no good)	Quick to judge app (and to leave if no good)
Willingness to wait	Want instant gratification	Limited patience
Back button	Not used (young kids) Relied on (older kids)	Relied on
Reading	Not at all (youngest kids) Tentative (young kids) Scanning (older kids)	Scanning
Physical limitations	Slow typists	None (unless disabled)
Scrolling	Avoid (young kids) Some (older kids)	Some
Animation and sound	Liked	Usually disliked
Advertising and promotions	Can't distinguish from real content	Ads avoided (banner blindness), promos viewed skeptically

An interesting observation we made was that kids suffer from a learned path bias: they tend to reuse the same method they've used before to initiate an action. We often saw kids who had been successful with a certain approach to an app stick determinedly to that approach over and over again, even as it failed them during subsequent tasks that required them to use a different navigation scheme.

2. Designing the user interface

2.1 User Interface Features

One of the first questions we had to answer was “what features do we add in our new app?”. To answer that we came up with some core principles we were going to follow.

The app should do just one thing really well.

We needed to identify the ONE killer feature in our app and only develop that. Any additional features should not be developed unless they are required. For us that feature was easy navigation inside Android for kids.

Quick path to value.

The app should demonstrate its value as soon as easily and as quickly as possible. Unless the user experiences the value of the app there is a large possibility that the user will get busy with something else and never come back to our app.

Instead of predefining a long list of specific features we chose to design iteratively, conducting many rounds of quick user testing and adding features as we go.

2.2 User Interface Design

It has long been recognized that user interfaces should be designed iteratively in almost all cases because it is virtually impossible to design a user interface that has no usability problems from the start. Typically, we completed a design and noted the problems several test users have using it. These problems would then be fixed in a new iteration which should again be tested to ensure that the "fixes" did indeed solve the problems and to find any new usability problems introduced by the changed design. During the user interface design phase we set the following principles to better achieve our goals.

Design a Super Clear UX.

Designing a good UX for adults often involve putting in bumper lanes and holding their hand: you want to make navigating your app very clear with little room for error. When it comes to designing an app for children, we basically had to triple these efforts. Kids between 6 and 8 years old have less developed motor skills, so we relied on big visuals and a larger hit detection area for actions, as well as limiting the available actions on screen.

Design for Impatience.

Designing for impatience was a key part of the process. Not only are the icons big and the visuals easy to understand, menus were hidden to minimize accidental touches and navigation was streamlined and simplified to help children easily follow the program.

Give Them a Breather Every Couple Hours.

We always want to encourage kids to get offline and play. Not only does this help rest their eyes, but allow for interactions with the physical world which is turning into a precious commodity in our increasingly digital reality. Our app prompt every couple hours to take a break. Also there are build in settings for the parents to set a timer.

A “childproof” navigation.

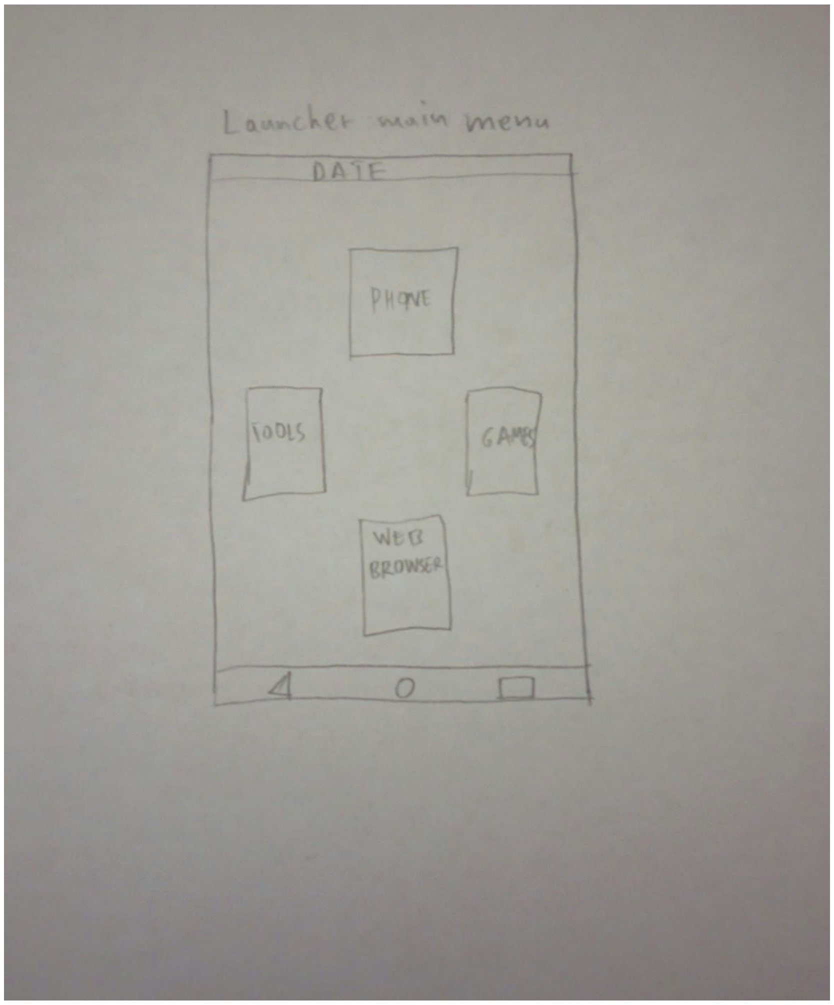
A serious usability problem happens when a young user inadvertently touches a menu button. For inquisitive kids, a menu button visible on all screens can sometimes be a source for frustration. A "childproof" navigation interface was important in order to prevent children accidentally brushing or touching the menu while using the application. The idea was simple: design menu access for the parent only. Our app requires someone to press the volume down button 3 times to activate the menu. Just like a childproof safety cap for a medicine bottle or other devices made safe for young children, we created a menu system that is childproof.

The following images are some early prototypes we created before we started the development in order to get feedback from the users.

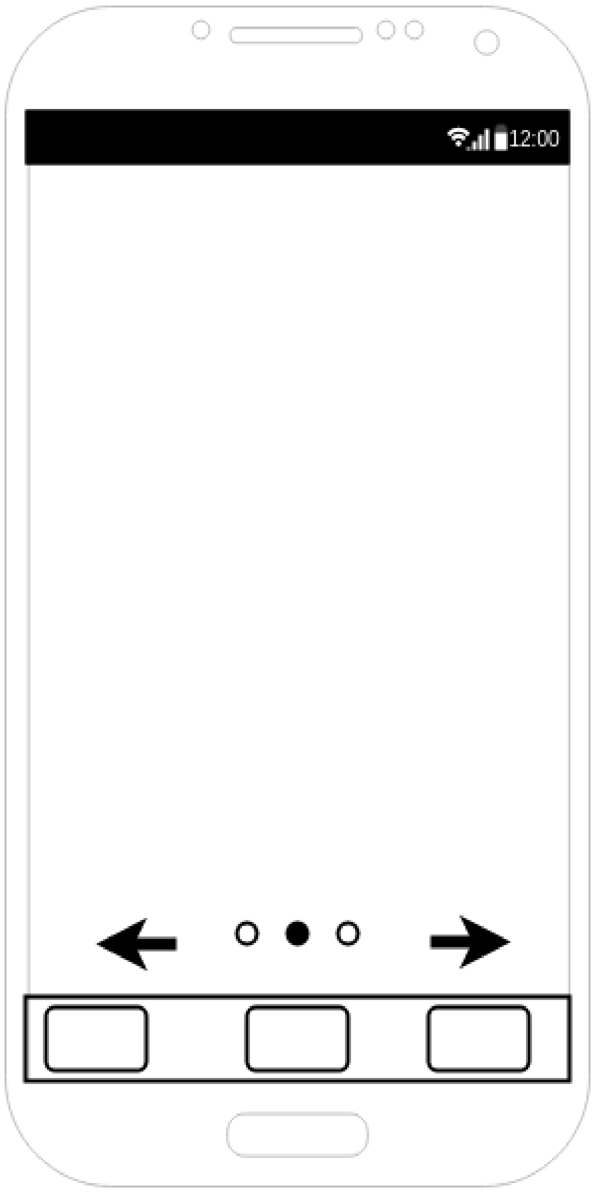
2.3 User Interface Development

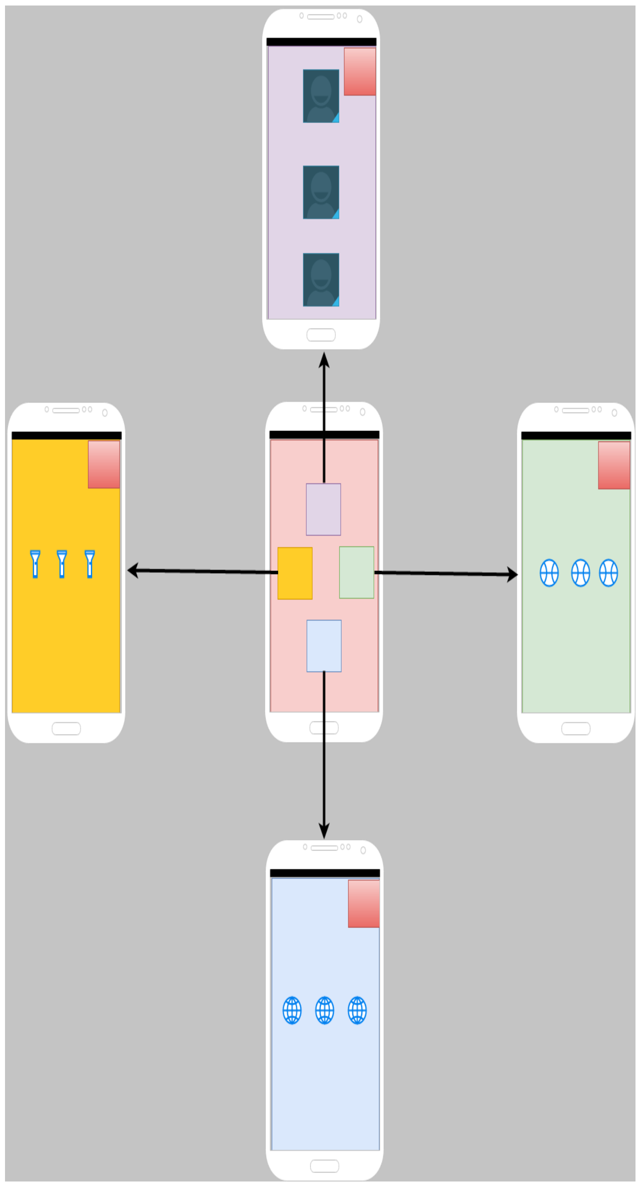
There are several ways to develop software, two of the most prominent methods being waterfall and Agile. We chose to go with agile as we thought it was a better fit for this specific project. Agile is a development process that focuses on iterative development, requirements and solutions evolve and adapt over time. We thought of design as evolutionary throughout the process, rather than a plan that we needed to stick to. When each feature was demonstrable user acceptance testing was done. We tried to have several shorter interactions with users rather fewer longer interactions. In general, Agile techniques work well for non-trivial user interfaces. However, there are many useful techniques we followed from other disciplines that can add significant value. Specifically, techniques from interaction design helped us with designing and testing user interface ideas before investing time in coding:

- By using **personas and goals**, we got an early feel for what kinds of tradeoffs might be necessary when designing the app.
- **Prototyping techniques** ranging from paper prototyping to interactive electronic prototypes can provide a means of rapidly iterating and testing user interfaces.



Early paper prototype of the main menu





Early prototypes of the main menu

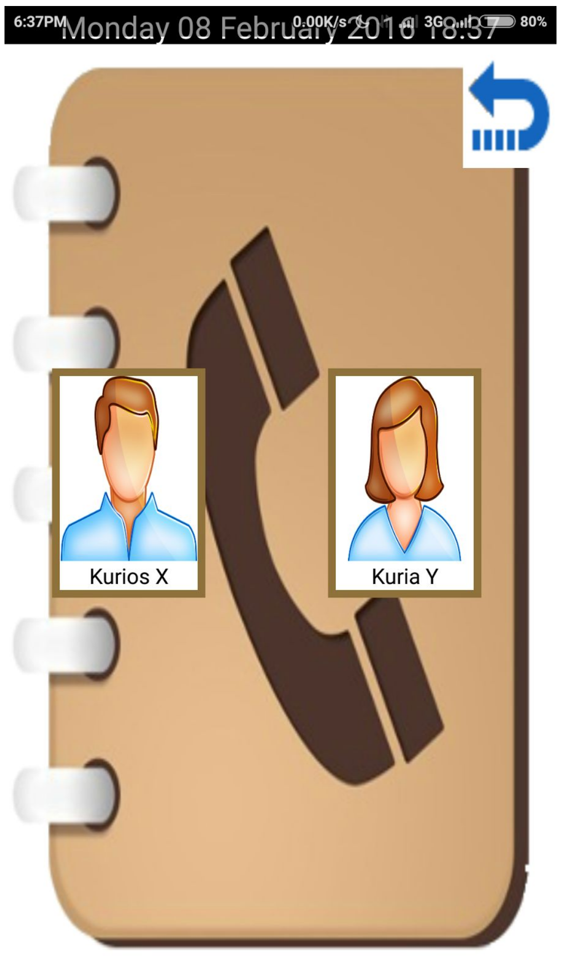
- Finally, **spike solutions** can be helpful when the user interface needs of a project are more complex than simple prototypes can communicate. By implementing a small slice of functionality that demonstrates a key user interface concept, a spike helped us ensure clear communication with users.

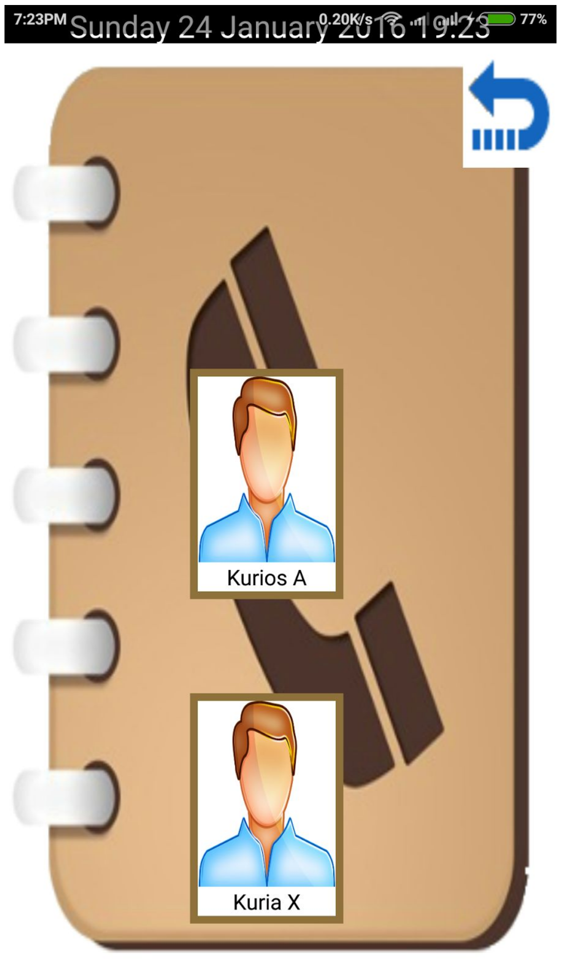
3. Evaluating the application

3.1 Evaluate User Experience

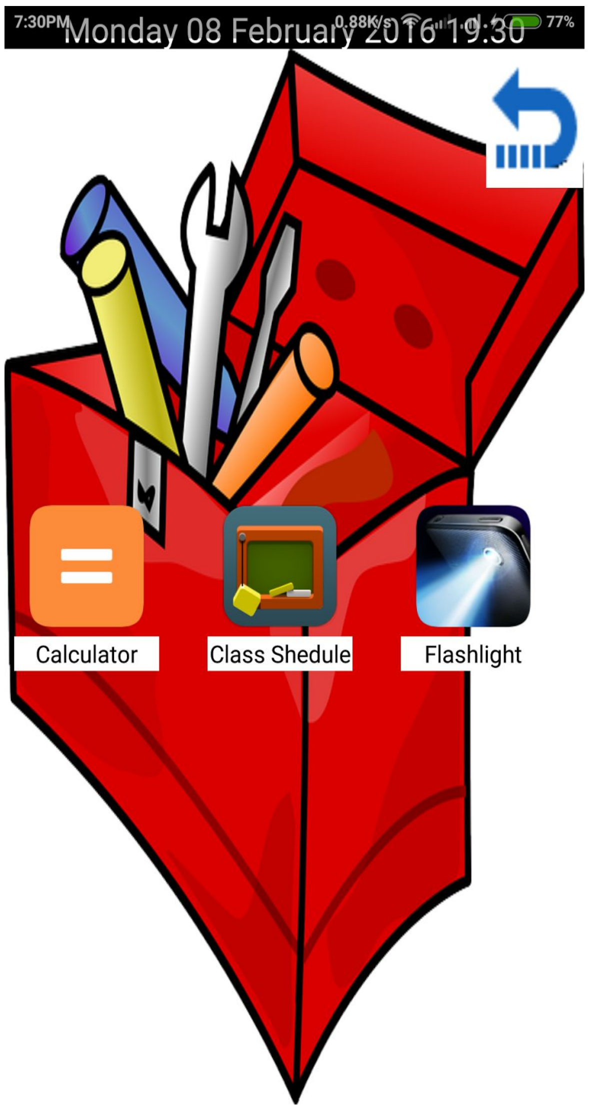
We tried to evaluate the user experience extensively both in the early stages of designing the user interface and during development. We sat down with the users to observe how they interacted with the user interface, we got feedback about things they didn't like and asked them about features they would like to see. We used a/b testing extensively by providing different users with different versions of the app in order to decide which design style to follow. Finally we let users decide even for the little details like the icons or the font style.

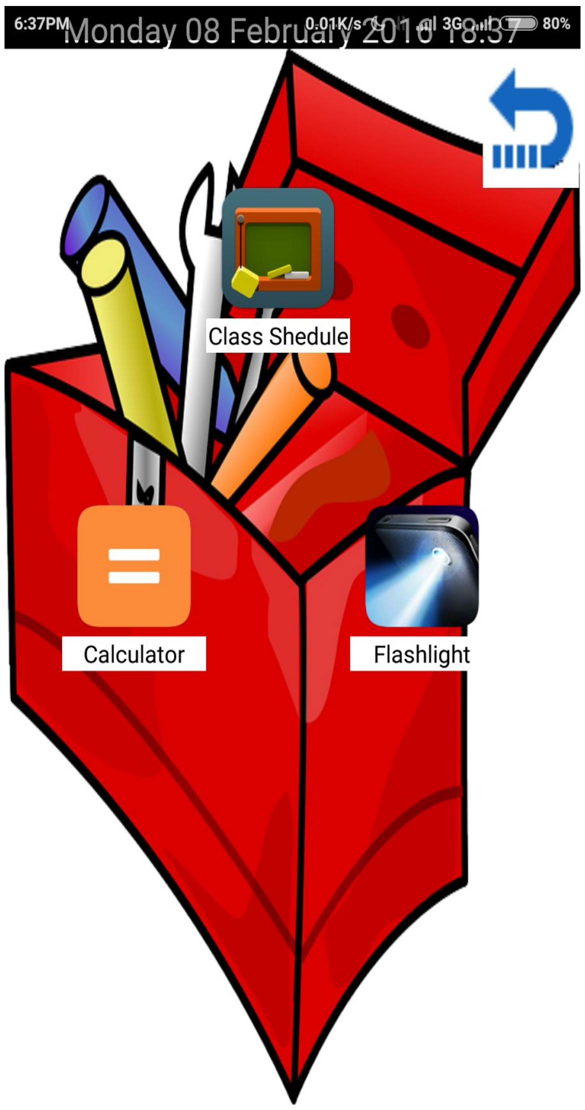
The following images are examples of testing different layout and getting feedback from the users about which they liked better.





The users chose the left one.





The users chose the right one

3.2 Conclusion

Kids are different, and so are their apps. We faced the unique challenges inherent in designing apps for children, and strategies for creating apps that are fun, educational and usable. As a team we established some cooperation principles. Each change was preceded by extensive discussion. Each member presented his opinion about how the change should be designed and implemented. Disagreement was rare, but in case there was one, we would always reach a fruitful comprise by reminding each other the fundamental software design principles. Finally, acknowledging each other's strengths and weaknesses during the semester, not only enhanced our contribution quality, but also provided breeding ground for individual intellectual development throughout the activity.