Master's Thesis

Czech Technical University in Prague

F8

Faculty of Information Technology
Katedra teoretické informatiky

Tablet infotainment system

Bc. Michael Bláha

January 2016
Supervisor: Ing Jan Šedivý, CSc.

/ Declaration

Abstrakt / Abstract

Tento dokument je pouze pro potřeby testování.

This document is for testing purpose only.

Contents /

Chapter 1
Introduction

sources:

- https://docs.google.com/document/d/1pGtlS5uY4PdKfHjf83dFGrajyVcea0tvzAnwXf61Cs8/
  edit
- generally progress: https://docs.google.com/document/d/1CEWym7MphsCO0v3CXe_bTHOgFBgquBbPVb
  edit

1.1 Project

See above

1.1.1 Motivation

See above

1.2 Assignment analysis

1.2.1 Assignment tasks

1.2.1.1 Review existing Android applications for in-car use

One of the key approches in research project is reviewing the existing progress in the given field. Reviewing existing applications helps understanding the topic, seeing the bigger picture, learning from mistakes of others and last but not least, getting general idea about competition.

1.2.1.2 Review and analyse User Interface development methods for in-car infotainment applications

Cosindering the car environment, the user interface must deal with a lot of different problems than usual. This task should review existing User Interface development rules and apply them to the car environment, then analyse them and choose proper method for car-UI design process.

1.2.1.3 Analyze the in-car OBD API and exported data

On-Board Diagnostics API is a standard API provided by modern cars for gathering various information from speed to engine temperature. This task focuses on understanding and gathering data from the OBD API.

1.2.1.4 Design an application system architecture for accessing the OBD data and resources

Having the data from OBD and preparing an application for displaying them, designing proper architecture is required for everything to work well. The application has to gather data, while displaying them properly without unneccesary (FIX!) delay.

1.2.1.5 Design a tablet User Interface for in-car use

After reviewing existing applications and UI development methods, the next goal is to create new User Interface for in-car use, while considering the constraints this environment puts on it.

1.2.1.6 Design and implement in-car application offering the OBD data for Android tablet platform

With everything prepared and thought through, the application will be developed based on result from all the tasks accomplished so far. In this case, the Android platform will be used as explained later in the text.

1.2.1.7 Perform UI and application testing and evaluate results

For best results the application must and will be tested. Both code and UI must be tested properly, using various testing approaches, such as unit tests or UI testing with reaul users in a car simulator.

Chapter 2
Analysis

sources:

- application analysis https://docs.google.com/document/d/1QyOiMzV0ikcDhPY3P5MsRL_80cCGzjoGf
  UL0/edit
- priority list https://docs.google.com/document/d/1juKYgUUDSI5CmfzjR4BsYSPHVYCGqrWuejgbhqzw
  edit

2.1 Existing applications

2.1.1 Applications

sources:

- https://docs.google.com/document/d/1p_pSGTUHEojOyP7ICCDNVV7RW1vn8iN_KECipC4Y9tY/
  edit
- http://www.makeuseof.com/tag/5-best-dashboard-car-mode-apps-android-
  compared/

2.1.2 Torque

Starting with an empty screen, lot of settings are required before using this application, since there is no default mode. Adding new views is easy and intuitive, but still very confusing. The add menu lacks hierarchy and everything is just sorted array of various options. There is no cancel button when popping the menu dialog.

This application can actually show almost anything OBD provides. It supports differents types of display, but it is hard to tell by their names. Responsiveness it not smooth at all and launching the application in horizontal mode confuses it, everything behaves like if it was in vertical mode.

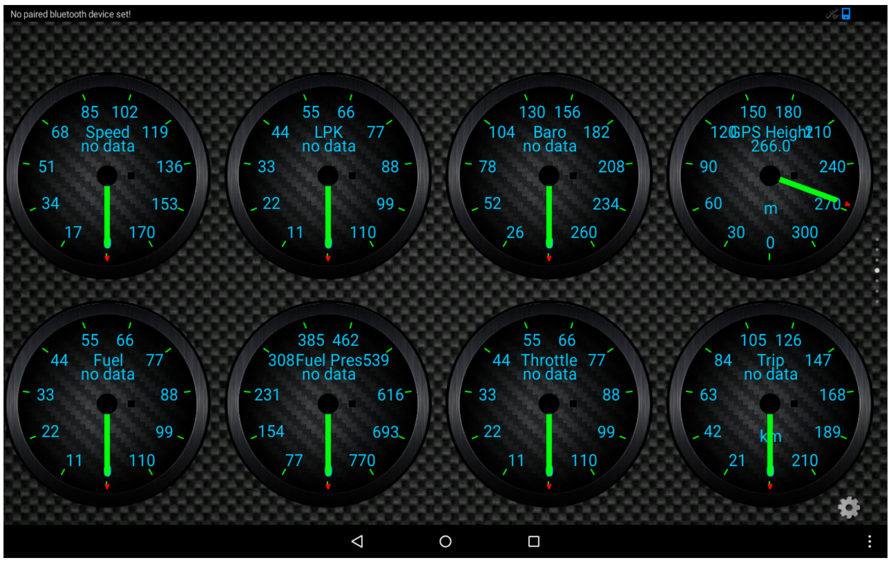

Figure 2.1. Screenshot from Torque

2.1.2.1 Advantages

- Lot of data from OBD available,
- various layout settings and themes,
- HUD mode.

2.1.2.2 Disadvantages

- One-level confusing menu without hierarchy,
- limited size options for displays (3 types),
- lacks default mode with predefined displays,
- hard to place displays, the grid does not work well,
- slow and laggy.

2.1.3 CarHome Ultra

This application starts with a pop-up tutorial for it's elementary functionality, telling the user about the speedmeter, compass, weather forecast and customizable dashboard for launching external applications. In default it offers Google Maps, Google Navigation and voice search. Adding another external application shortcut is done by tapping the tab. Also there are basic settings, which offer brightness mode (day, night, auto), theme and safety options.

It appears to be just a simple application offering speed, compass, weather and external application launcher. The new version also displays location (address) and a phone version is able to respond to text messages. It also supports text to speech (on touch).

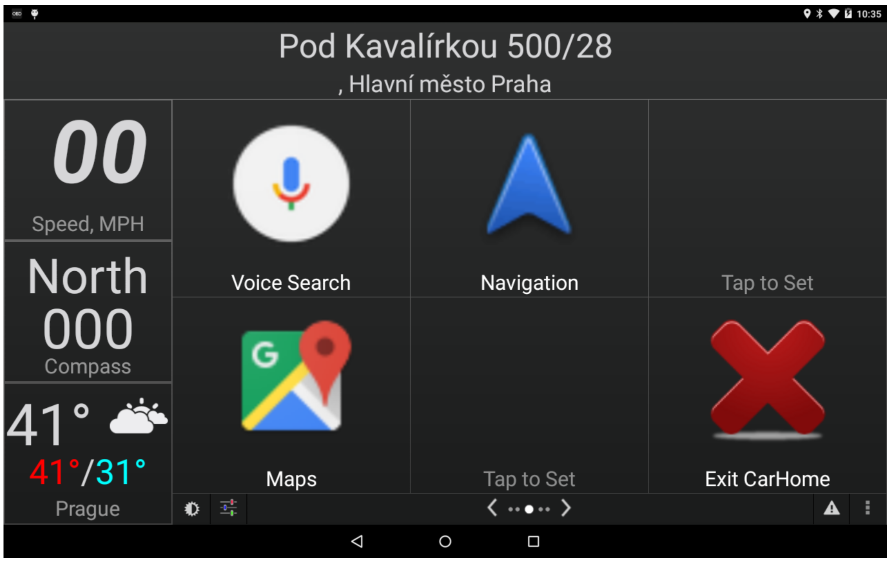

Figure 2.2. Screenshot from CarHome Ultra

2.1.3.1 Advantages

- Simple UI, easy to understand,
- responsive, fluent,
- possible to change units (mile/km, etc.),
- lot of themes available,
- adjustable update rates,
- a lot of different settings.

2.1.3.2 Disadvantages

- Small buttons on small screens (fixed 6 buttons),
- even smaller setting buttons
- limited functionality
- tapping weather makes the app speak for every single tap, no matter if it already speaks (it can speak for hours after few taps).

2.1.4 Car Dashdroid

After a long loading the main window appears. It has three screens, which change by swiping right or left. The left screen contains dial keyboard, the right screen contains customizable cards (for external application shortcuts or built-in tools) and the main screen consists of weather, speed and shortcuts to contacts, music, navigation and voice command.

It also provides settings for bluetooth, brightness, screen rotation, fullscreen, day/night mode and application settings, where other options can be set, such as home adress, theme, units.

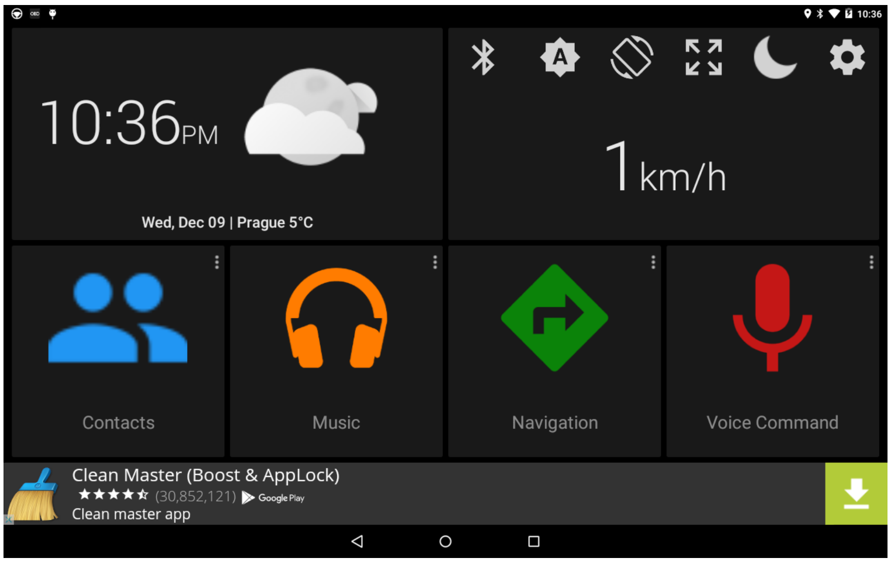

Figure 2.3. Screenshot from Car Dashdroid

2.1.4.1 Advantages

- Simple UI, easy to understand,
- responsive, fluent,
- possible to change units (mile/km, etc.),
- able to read incoming SMS using TTS.

2.1.4.2 Disadvantages

- Very limited functionality
- not optimized for tablet,
- distractive commercial ads in free version.

2.1.5 Ultimate Car Dock

While the design is very similar to CarHome Ultra, this application offers fewer displays on a single screen. There are five screens, each one consists of six cards. Every card can change into shortcut or a build-in application. The Ultimate Car Dock has only few built-in applications: music player, voice command, speed, weather, messages and calls. It also supports shortcuts to other external applications.

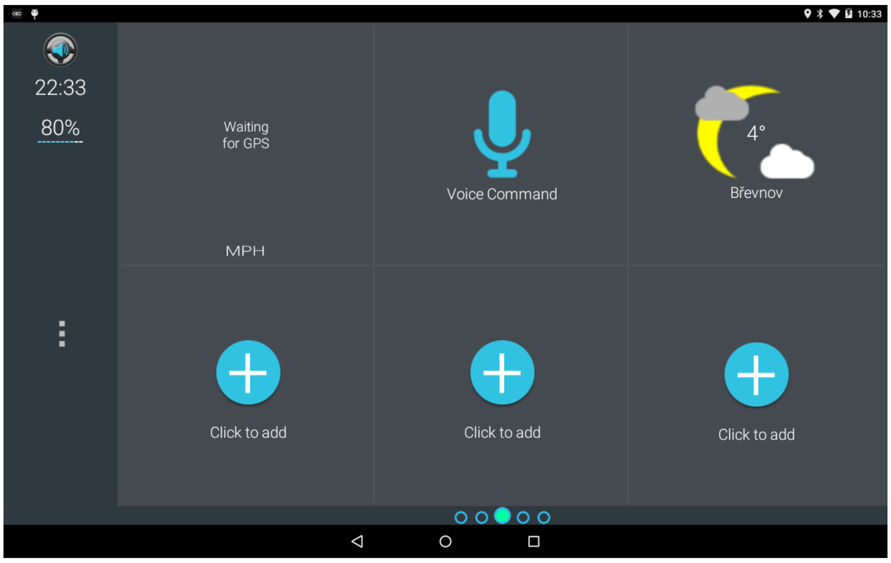

Figure 2.4. Screenshot from Ultimate Car Dock

2.1.5.1 Advantages

- Simple UI, easy to understand,
- responsive, fluent,
- possible to change units (mile/km, etc.),
- able to read various incoming notifications using TTS (Gmail, WhatsApp, etc.),
- predefined SMS responses (selectable when a message comes),
- direct calls and messages (shortcut to call/message a certain person).

2.1.5.2 Disadvantages

- Limited functionality
- not optimized for tablet,
- small font.

2.1.6 Conclusion

Except by Torque, which focuses mainly (and only) on OBD, all the applications are very similar to each other. They have similar design and functionality – mostly weather, speed provided by GPS, voice command and shortcuts for external applications.

2.1.6.1 Suggestions

- OBD data,
- shortcuts to other applications,
- adjustable cards,
- built-in cards (weather, speed, voice command, etc.),
- simple grid UI,
- possibility to change displayed units,
- responsive and fluent,
- day and night theme,
- predefined message and call responses,
- TTS for incoming notifications.

2.1.6.2 Possible issues to avoid

- Responsiveness,
- limited functionality,
- small and hardly visible font,
- distractive ads.

2.1.7 Android Auto

sources:

- `http://developer.android.com/design/auto/index.html`
- `https://www.google.com/design/spec-auto/designing-for-android-auto/designing-for-cars.html`

Recently, Google presented new application model for information delivery while driving. It is called Android Auto, it provides a standardized user interface and user interaction model for Android devices. Focusing on minimazing the driver distraction, it presents a few options to interact with user. It supports three application types:

- System overview
- Audio applications
- Messaging applications

2.1.7.1 System overview

System overview is supposed to be a home screen for Android Auto application. It presents both current and past notifications. The amount of notifications is limited based on screen size. Every notification consists of an intent icon, text and image, while following certain sizing rules. Every such notification can be expanded on the spot or another subapplication can be launched.

2.1.7.2 Audio applications

Audio applications in Android Auto have a simple template structure. It consists of a main consumption view, a drawer and a queue screen. The main consumption view displays a few control elements and a cover background. The drawer is a simple list and provides access to favorite and popular content. Finally the queue screen displays a list of pending content, for example songs in a queue.

2.1.7.3 Messaging applications

Focusing on minimizing the cognitive load, messaging concept in Android Auto focuses on voice control over looking and typing. It allows reading the message outloud and responding with a set of predefined voice commands as well as dictating a whole message using built-in speech recognition.

2.1.7.4 Conclusion

It seems to be a good sign that even Google is interested in this area and performs such a research. Every Android application can be designed for Android Auto and use it's simplified user interface, allowing the developer to focus on other issues than in-car user interaction. However, the functionality is currently very limited. Hopefully there will be further progress as soon as possible.

2.2 Platforms

Possibilities: Android, iOS, WP(?) State requirements, properties, criteria

The chosen platform heavily influences the piece of market an application can reach. Therefore, only platforms with solid market share are considered. Another criteria is the simplicity of development, which influences the time and effort put into an application before it can be released. This is especially important for quickly finding out the sale potential of an application. Following the first rule mentioned above and based on tablet sales in past years (sources: `http://techcrunch.com/2014/03/03/gartner-195m-tablets-sold-in-2013-android-grabs-top-spot-from-ipad-with-62-share/`), the only viable options for an application are platforms Android, iOS and Windows.

2.2.1 Android

In 2013, the Android platform had 61.9 % market share, making it the most used platform in the world. Targeting the Android platform would create large base of potential customers.

The development language for Android is Java, commonly known object-oriented programming language with solid developer base. Therefore it is easy to find developers as well as answers to variety of programming related issues, making the development much easier.

2.2.2 iOS

With 36 % market share in 2013, iOS is the second most popular tablet platform. Considering a typical iOS user, who is willing to pay for quality, iOS could be a good choice for an application in context of potential customers.

However, the development language called Swift is somewhat new in the world, which brings a lot of possible difficulties. Searching for answers while developing in this technology might prove too troublesome.

2.2.3 Windows

With only 2.1 % market share in 2014, the Windows platform does not seem to be a valid choise for given criteria. Having thirty times lower customer base than Android, it goes into the nice-to-have section when it comes to multi-platform applications.

2.2.4 Conclusion

Fulfiling the requirements for customer base as well as simplicity of development, the Android platform seems to be the best choice available at the time of writing this. As such, it will be analyzed more thoroughly later in this chapter.

2.3 Android platform

2.3.1 Architecture

2.3.2 Specifics

2.4 GUI

2.4.1 Basic principles

sources:

- MI-NUR `https://edux.fit.cvut.cz/courses/MI-NUR/lectures/start`
- Designing for indash automotive `http://revinity.com/?p=128`
- UX design stackexchange `http://ux.stackexchange.com/questions/51968/what-ux-guidelines-should-one-keep-in-mind-when-designing-the-gui-for-a-automobi`

2.4.1.1 Consistence

Dont make user learn things twice

2.4.1.2 Simplicity of usage

KIS principle

2.4.1.3 Shortness of learning curve

Easy to learn, critical environment

2.4.2 Car UI differences

What else to consider?

2.4.2.1 Controls

Sizes, big enough to touch

2.4.2.2 Visibility

Sizes, fonts big enough to see, data

2.4.2.3 Contrast

Colors, visibility at night/day

2.4.2.4 Responsivness

Responsive, must see whats going an as fast as possible

2.5 Development and support tools

2.5.1 Development environment

2.5.2 Quality Assurance tools

2.5.3 Version system

2.5.4 Test driven development

2.5.5 Continuous integration

Chapter 3
Design

3.1 Application architecture

3.1.1 Extensibility

3.1.2 Modularity

3.1.3 Adaptability

3.1.4 AutoUI preparation

3.1.5 Platform limitations

3.2 GUI

3.2.1 Basic elements

Basic idea

3.2.2 UI drafts

Describe the process, phases, analyse and compare advantages, disadvantages, thoughts

Chapter 4 Realization

4.1 Preparation

4.1.1 Environment

4.1.2 Versioning

4.1.3 Testing

4.1.4 Scripting

4.2 Core

4.2.1 Core

4.2.2 Data storage

4.2.3 Communication

4.2.4 Optimization

4.3 Modularity

4.3.1 Requirements

4.3.2 Integration

4.4 GUI

GUI implementation based on the design! Implementing modules, color, responsive effects

4.4.1 Common elements

Hierarchical model, effects, submenus

4.4.2 Multiple designs

Limited set of module types

Chapter 5
Testing

Brag about TDD, CI and Simulator!

5.1 Code

Describe testing code, common testing (look&see, etc.)

5.1.1 Unit testing

Unit testing on android, mention Test driven development, continuous integration, automatic tests, consider giving an example

5.1.2 Integration testing

Instrumentation? Describe TDD, CI, automation

5.1.3 System testing

Server testing, consider removing

5.1.4 Qualification testing

Testing with users - consider section on its own - testing application as a whole thing

5.2 GUI

5.2.1 Heuristic testing

Introduction, description

5.2.1.1 Evaluation

sources:

- `https://docs.google.com/document/d/1LAPqmYqe5LBE6vqWpi-rRYjHY1-zVPCzkFP2Gvh5i-Q/edit`

5.2.1.2 Conclusion

Did not have time to fix

5.2.2 Testing with users

5.2.2.1 Usability testing

Testing the application as a regular application. Is it understandable? Is it easy to control, to see data, to understand, to comprehend, to learn?

5.2.2.2 Simulator

Describe the car simulator in Albertov. DO NOT FORGET TO THANK THE DEPARTMENT OF DRIVING SMTHING, CVUT FD

5.2.2.3 Preparations

Selecting the world models and preparing them for testing, installing EyeTracker cameras, installing WebCamera, preparing data gathering, designing scenarios

5.2.2.4 Course

The testing itself, describing participants

5.2.2.5 Evaluation

Evaluating results

5.3 Summary

Chapter 6
Conclusion

6.1 Assignment completion

6.2 Project life cycle

6.2.1 Present

6.2.2 Future

6.3 Summary