TECHNICAL DESCRIPTION FEEDME

Authors

Name	*Position*	*Email*
Vendela Palmberg	Technical Writer	venpa358@student.liu.se
Gustaf Teneberg	R&D Manager	guste822@student.liu.se

Updates

Sprint	*Version*	*Date*	*Modification*	*Person in charge*	*Verified by*
3	0.1	2014-11-27	A short first draft, just some points of what we colud include	Vendela Palmberg	Gustaf Teneberg
	0.2	2014-12-01	Review of the earlier version with some changes and elaborations in the first part. Also added a more thorough part about the library.	Gustaf Teneberg	Vendela Palmberg
	0.8	2014-12-03	proof read	Vendela Palmberg	Ylva Johansson
	0.9	2014-12-04	Added help controller and help view	Vendela Palmberg	Ylva Johansson
	1.0	2014-12-09	Updated after inspection	Vendela Palmberg	Jakob Boman

Preamble

This is a brief description of the structure and flow of the calls in the code to read before checking comments and code, this to give a general understanding of which function is called where and in what order. For detailed descriptions of all functions, either view the comments in the code or go to the GitHub application workspace.

Structure of the code

Main app

The main app consists of two different views, one view for showing a list (the feed) of articles and one view for showing the article itself. The program is built around two main classes; one view class with some basic functionality for a view that is inherited to classes for each view (main_view, article_view and help_view). The other class is a controller class, which is also inherited by three subclasses, each representing one view (main_controller, article_controller and help_controller). There is also a news_model class that manages the news from the rss-parser. The objective of the controllers is to control the flow of data and logic of the app. In the controller class there are functionality controlling the navigation through the input from the remote and the communication between the news_model and the views. The views only purpose is to print the graphics that the controller tells it to print. No logical operations are done here.

Library

Apart from the main app there are some help modules. First of all there is a graphics module that helps print the graphics on the computer, a so-called emulator, with the use of the Set-top-box API. In the library there is also a font loader module and a text printer module for converting strings to images and printing them to the background, which works both on the emulator and the Set-Top-Box. The font loader module is used to load the fonts at the beginning of the app starting. And the text printer is a fully functional tool to convert the headlines, ingresses and article contents to images with customized width, height or alignment on the surface. In the library there is also an xml to table module that parses an rss file into a table. The parsing function has functionality that fetches certain parts of the rss file and adds them into a table. For each feed in the rss file there is a tuple in the table containing the headline, description, publication date, link, image link and category. In the library there is also a download module that fetches articles and rss files from a server using the new API with IP support.

Flow of calls in the code

The flow of the calls in the code are managed by the two controllers based on what view you are currently showing. The flow will now be described first of what is happening in the main view and second the article view.

Main view

- Loaded on start
- Fetched news from **news_model:get_news(feed)** that retrieves news from the xml_to_table module with function **read_xml()**
- Create a **main_view:new()**
- Calls **load(arguments)** in main_ view that prints the headlines of the news, the thumbnail, the background and the highlight
- Navigation (up, down, left, right): calls function **load()** and reprints everything (headline at another position)
- Press menu: **open_main_view()** is called
- Navigate between feeds to the right: **open_next_feed()**
- Navigate between feeds to the left: **open_previous_feed()**
- Navigate to a new page to the left: **has_left_news()** called
- Navigate to a new page to the right: **has_right_news()** called
- Press ok when highlighting an article: load article_controller with function **load_controller(Article_controller)**
- **Press info button i:** load help_controller with function **load_controller(Help_controller)**

Article view

- Fetched the article from input arguments: feed, position and page number with function **retrive_article()** which calls **get_article()** in news_model that calls function **get_news_feed()** in the same module.
- Creates view with **article_view:new()**
- Show all the content of an article by calling function **show_article()**
- **show_article()** calls functions **get_content()**, **get_ingress_from_content()** and **get_text_from_content()** to separate the ingress from the article text in the content
- **show_article()** then calls the functions **print_background()**, **print_category()**, **print_article_text()**, **print_date()**, **print_title()**, **print_ingress()** and **print_image()** that all calls the function **show_image()** in the graphics module
- Navigation: when moving left and right the function **retrive_article()** is called again where a new article is fetched. The **show_article()** is then called again.
- When pressing the menu button the main_controller is reloaded with function **load_controller()**.

Limitations

The rss parser in the xml_to_table module only works if the xml file is written in the same way as the current xml files. It does not support 'CDATA' sections or other formats that the current files doesn't have. The current xml files are taken from CNNs rss feed.

The articles can only be a maximum of approximately 170 words (depending on length or words and number of text sections.

Running the code on the box

In the source code a "REEDME.txt" file can be found which explains how to make the app run on the code on the set top box. The following information is presented in that file:

To run the app on the box:
1, To the USB move: app.lua, lib and src
2, Change row 5 in app.lua, PATH variable, to the current USB path
3, Run file app.lua