FreeType 엔진에 새로운 폰트 서비스를 추가하기 위한 컴포넌트 모듈 설계 및 구현

아마르 울 하산, 김성민, 최재영
숭실대학교 컴퓨터학과
e-mail: arrmar.instantsoft@gmail.com, sungmin.kim@ssu.ac.kr, choi@ssu.ac.kr

FreeType Outlet Adapter for adding new components

Ammar Ul Hassan, Sungmin Kim, Jaeyoung Choi
School of Computer Science & Engineering, Soongsil University

Abstraction

FreeType is a rasterizer which is commonly used in different operating systems like Linux, android etc. Although FreeType is open source but it is not easy to add/remove modules and services etc. for developers. This paper proposes a new module for FreeType named as FreeType Outlet adapter (FOA). It enables to add/ remove modules, services, functionality etc. inside FreeType. It acts as the bridge to add functions from outside FreeType to the inner core of FreeType. New font formats like METAFONT, animated fonts and customized fonts which currently are not supported by FreeType can be added with this FOA module.

1. Introduction

The text is an effective method for gathering information and communicating with other people. Even if smart devices are commonly used these days with effective resources like media and sound, text plays the key role between user and device. Text is represented by characters, and these characters are physically made from specific font files in digital environment system. These font files are used in specific system module called rasterizer and FreeType [1] is one of the most famous rasterizer based on Linux operating system. Many kinds of systems using Linux platform include FreeType rasterizer for printing Text to various display devices such as android mobile operating system, Apple operating system, embedded Linux devices, etc.

FreeType is a system library which can rasterize only two kinds of font types, which are bitmap and vector fonts. Unfortunately, this rasterizer does not provide other types of font format and every driver modules are only bounded to FreeType. Even though FreeType is open source but it is not easy to add/remove new type of drivers inside FreeType after it has been build. In addition, when some application prints text on Linux system using FreeType, it needs some additional libraries. There are mainly two libraries used, font config [2] and Xft [3]. Fontconfig library is used for managing font files in the Linux system, and Xft library provides API for printing text for application through the FreeType system. Although these libraries are working together with FreeType system, and are strongly connected with each other but still these are installed independently.

This research suggests a FreeType module for adding/removing new functionality easily from outside of FreeType. This new module is called FreeType Outlet Adapter (FOA), which will add/remove modules in FreeType. This module can provide interfaces for adding/removing new functions, so it can be used by various available services mentioned in <Table 1>. It can be used by unique type of font files like METAFONT [4], customized font format files [5], and animated fonts [6], etc. It can automate font style functions and manage font files such as fixed margin [7] of characters, fixed fonts size [8], and font file management systems [9]. At last, it can also be used for data filtering like METADATA filtering [10] in font files.

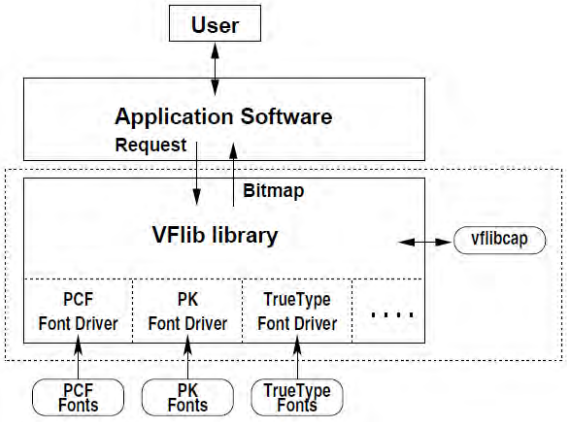
<Table 1> Types of services about font files

Type	Available Service examples
New font types	Programmable font (METAFONT) [4] Customized font [5], Animated font [6]
Fixed font style	Automated margin of characters [7] Automated font height and position [8]
Management	Font file management [9]
Data filtering	METADATA in font file [10]
Etc.	Etc.

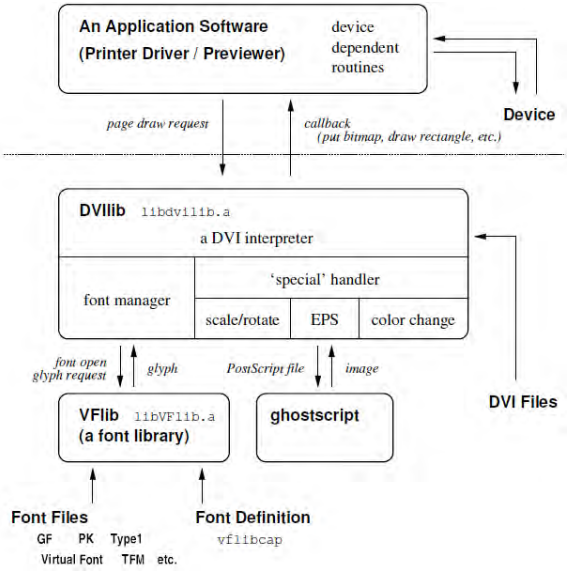
2. Researches related with rasterizer

There are some researches on rasterizer, especially based on Linux. VFlib [11] [12] framework library can add/remove driver modules conveniently and link them to rasterizer. It is possible to use font driver interface in VFlib itself as shown (Figure 1). Application software can approach VFlib easily using VFlib API, too. But this framework is too heavy for embedded systems because it includes too many default metadata which can't be removed.

DVILib [13] is used for printing DVI files to print text based on VFlib. This research shows that text can be printed by uncommonly used font formats. It means that another font



(Figure 1) VFlib architecture



(Figure 2) DVILib based on VFLib

format other than bitmap and vector can be used as new font type in some rasterizers. But DVILib fitted only in VFlib, and it has same problems like VFlib.

Another research, MFCONFIG [14] module added new functions in FreeType system. It can support programmable font METAFONT, but this module depends on fontconfig library only. If there is a FreeType system without fontconfig library, MFCONFIG module can't be used there and this library works very slow for some tasks like converting fonts, whenever styled text is printed.

General researches based on rasterizer are strongly dependent on that rasterizer itself. No rasterizer provides a convenient approach for adding various functionalities in it. If there is a rasterizer which can provide some plug-in environment than many additional functionality can be added in that rasterizer. So, this research suggests a module for adding functionality conveniently for FreeType rasterizer. This module can easily be used in any system using FreeType, and new services can easily be added/removed inside FreeType. This module is named as FreeType Outlet Adapter in short called as FOA and with this module new types of modules, functions etc. can be added in FreeType. This module consists of four sub modules which can link the functionality existing outside of FreeType to inner core of FreeType.

3. FreeType Outlet Adapter (FOA)

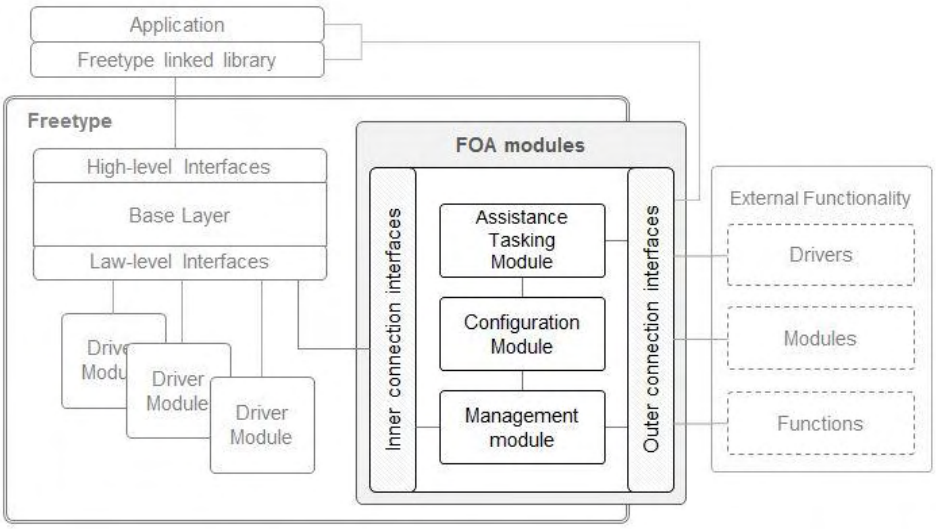
FOA consists of Communication module, Management module, Configuration module, Assistant Tasking module, as shown in (Figure 1). Communication module provides interfaces to external functionality which is to be added from outside and it connects FreeType inner modules to them. Management module performs the main core tasks in FOA module. It can manage added functionality and link them to FreeType inner modules. Configuration module is for registering or unregistering new functionality such as drivers, functions, and other modules. Assistant tasking module is in charge of optional functions like timer, setting values, scanning system etc. to help management module. These four main modules are explained below.

Communication module connects external functionality to FreeType through FOA. This module is divided in two parts: Inner connection interface and Outer connection interface. Inner connection interface is made by default FreeType interfaces like other inner driver modules. FOA module can be set up with FreeType through this interface. On the other hand, outer connection interface provides interfaces for new external functionality outside of FreeType. External functionality can be plugged in FOA using outer connection interface.

Management module is a main core program for operating FOA modules. It takes request of registering external functionality through outer connection interface. It also takes request of what service is needed from FreeType through inner connection interface. This module can find requested functionality which are registered in FOA, and can link related functionality outside of FreeType through outer connection interface. If some registered functionalities are passed to assistant tasking module, management module takes them and works on them. Otherwise, if functionality is needed to be removed from FOA, management module can remove them using configuration module and remove related tasks in assistant tasking module.

Configuration module works to register functionality in FOA. This module has private contents list inside, and it manages these lists. If management module requests to add new functionality in FOA, configuration module adds this new functionality in a private content list without duplicating same modules. If management module tries to call a service requested from FreeType, configuration module uses private contents list inside it and replies with the information which is stored inside it.

Assistant tasking module helps management module for various types of new functionality. It can reserve specific tasks when it works. If there is some functionality which needs to execute in a specific time, this module takes that functionality requirements. And it can send request message to management module for doing that respective work.



(Figure 3) Architecture of FreeType Outlet Adapter Module

4. Implementation of FreeType Outlet Adapter

FOA module implements four sub modules. Registering FOA module automatically registers all these sub modules. All of these modules are coded in C language, and have followed FreeType interface rules.

FreeType can be compiled in 2 ways: automatically, using the GNU make and manually. As FOA use FreeType interfaces like other driver modules, so it uses automatic way with configuration options. Four steps for registering FOA to FreeType are below:

1. Creating and Connecting sub modules inside FOA.
2. Creating Driver API specific interfaces, error handling and creating internal services, etc.
3. Creating Driver API specific implementation in Base layer of FreeType with step 2.
4. Registering FOA module in FreeType.

First, FOA module will implement four elements mentioned in chapter 3. Assistant Tasking module and Configuration module work only for FOA, so they are not needed to be registered in FreeType.

Management module implements with interfaces of FreeType and it can be divided in three parts. First part is making a link for error handling to FreeType. Every module in FreeType has to define two types of errors: prefix errors and module specific errors. Management module has these errors definition itself and can connect to FreeType through inner connection interface. Second part is specifying FOA module's information. Management module has all elements of FOA module like functions and interfaces for management. For registering in FreeType, management module defines some rules for making configurations. Third part is for connecting new functionality outside of FreeType. Like management module is connected to FreeType through inner connection interface, this module provides connection interface for new functionality such as registering, error handling defining through outer connection interface.

Management module operates configuration module and assistant tasking module. These modules are working inside FOA. Configuration module makes a list of functionality to be added in FOA, and manages this list for management module. Assistant tasking module makes a list of data for tasks that are reserved in management module. It has functions like alarm execution, making log files, callback events etc.

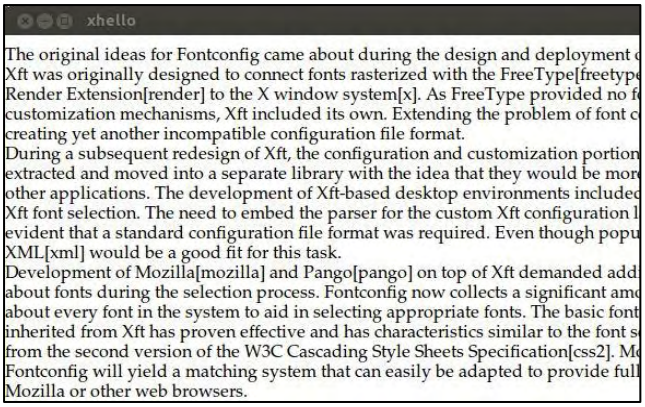
External functionality can easily be plugged in FOA module through outer connection interface. Every external function has to call management module for registering themselves with both information like what type of functionality (Module, service) and what error handling is needed. If contents need to reserve some specific action, then it can be executed by assistant tasking module.

5. Testing FOA module with external functionality

FreeType consists of different modules like Font Driver modules, Render Modules, Helper Modules etc. The Font Driver Modules are used to support different font formats. By default, FreeType comes with font drivers like Truetype, Type1, etc. These Driver modules are format specific i.e. they support only format specific to them like TrueType supports .TTF files.

For the testing of FOA module architecture, a new font driver was implemented in FreeType. This font driver is the replica of default font driver Type1. This Driver supports font format files like .PFA and .PFB. It has its own interface for client applications. It provides similar inner services like the original Type1 font driver. All the management related functionality is also similar to Type1 driver module like error handling specific to this driver, memory management implementation or to access font specific features etc. This driver is properly registered in FreeType. It can work with existing Type1 driver and can also work independently if original Type1 Driver is disabled from FreeType.

For the testing of this new Driver module an application which uses the X Windows System was developed. This client application is responsible for printing a styled text on screen. This client application connects all the Linux font system elements like Xft, fontconfig and FreeType. This



(Figure 4) Test Printing Text using new driver

application takes 2 inputs from user which is the font pattern i.e. font name and text on which font style should be applied. Fontconfig finds the font in the operating system using its font matching module and passes that to FreeType. FreeType than examines what kind of font format is passed and then calls its respective Font driver module.

For testing the new Font driver configured in FreeType a Type1 font of extension .PFA and .PFB were passed. FreeType called the new driver Module and applied all necessary functions and returned the rasterized text which was then showed on screen by using X windows system through Xft as shown in Figure 4. Different scenario testing was done like testing new driver module with original Type1 and testing new driver module without original Type1 driver.

This test showed that a new module inside FreeType can be implemented which can provide interfaces for client applications to easily add/remove different modules inside FreeType. The new Module can provide functionality like communicating with FreeType's inner core modules and also providing interfaces for outer client applications, managing modules inside FreeType with its own easy to use interfaces. Configuring new services and functionality inside FreeType and helping modules that can provide generic functionality to other modules. This new module can be the bridge for adding/removing different functionality inside FreeType by providing easy to use to use interfaces to client applications. This new module is named as FreeType Outlet Adapter FOA.

6. Conclusion

In this paper a new module for FreeType named FreeType Outlet Adapter is proposed. It enables to add/remove new functionality inside FreeType like adding/removing new modules. FOA can be used with various optional services like METAFONT, customized fonts, animated fonts, etc. The existing FreeType library can only rasterize two types of fonts which are bitmap and vector fonts. No new font format types can easily be added in FreeType.

There are some researches on Linux based rasterizers like VFLib library which manages to add/remove and link driver modules easily in rasterizer but this framework is too heavy for embedded system with limitations on memory.

FOA is a module for FreeType rasterizer which can provide plug-in environment through which new external functionality can be added/remove to/from FreeType. This module can easily be used in any system on Freetype.

FOA module is going to be researched more for linking various service contents and make interfaces like open code API. In this paper, one test example for making a font driver and plugging it in from outside of FreeType is described. If FOA module can become more detailed and can improve quality, then various font services shown in <TABLE 1> can be easily provided to users.

Acknowledges

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government (MSIP) (No. R0117-17-0001, Technology Development Project for Information, Communication, and Broadcast)

References

[1] David Turner, Robert Wilhelm, Werner Lemberg, "FreeType", www.freetype.org
[2] K Packard, "Fontconfig", Gnome User's and Developers European 2002
[3] Keith Packard, "The Xft Font Library: Architecture and Users Guid", XFree86 Technical Conference, Citeseer, 2001
[4] Donald E. Knuth, "Computers & Typesetting Volume C: The METAFONT book", 1986
[5] Benjamin P. Bauermeister, "METHOD AND SYSTEM FOR CREATING, SPECIFYING, AND GENERATING PARAMETRIC FONTS", US 5586241, 1996
[6] Cameron Bolitho Browne, "ANIMATED FONT CHARACTERS", US 6504545 B1, 2003
[7] Dov Harel, "METHODS AND SYSTEM FOR CONTROLLING INTERCHARACTER SPACING AS FONT SIZE AND RESOLUTION OF OUTPUT DEVICE VARY", US 5740456, 1998
[8] Nathan W. Everett, "INTERNATIONAL AUTOMATIC FONT SIZE SYSTEM AND METHOD", US 2004/0119714 A1, 2004
[9] Richard A. Gartland, "FONT NAVIGATION TOOL", US 6512531 B1, 2003
[10] Richard Hughes, "METHOD FOR EXTRACTING FONT METADATA FROM FONT FILES INTO SEARCHABLE METADATA FOR PACKAGE DISTRIBUTION", US 2011/0126191 A1, 2011
[11] H. Kakugawa, "VFlib- a general font library that supports multiple font formants ", EuroTEX conference, March 1998
[12] H.Kakugawa, "A general purpose font module for multilingual application programs", SP&E, March, 2000
[13] Hirotsugu Kakugawa, "A Device-Independent DVI Interpreter Library for Various Output Devices", TUG 2000
[14] Sungmin Kim, Jaeyoug Choi, Geunho Jung, "MFCONFIG: METAFONT plug-in module for FreeType rasterizer", TUG 2016