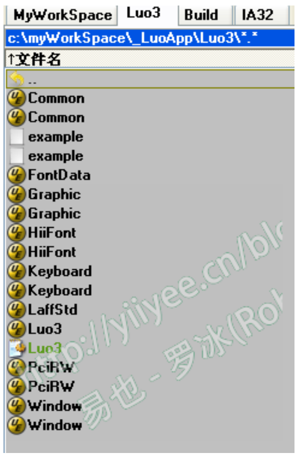
UEFI Development Exploration 19 – Using HII to Display Chinese Characters 4

原创 luobing4365 Posted on 2019-09-17 22:39:37 Read 956 times Collection 3 Likes copyright
Category columns: UEFI Development Article Tags: UEFI Programming UEFI HII UEFI color characters UEFI Chinese character programming Low-level programming

(Please keep it-> Author: Luo Bing https://blog.csdn.net/luobing4365)

The use of SimpleFont and Font in UEFI Shell has been implemented, and it is relatively easy to implement in Graphics mode.

Most of my work was spent organizing files and building source code files specifically for Hii according to the previous code structure, encapsulating some details for later use.



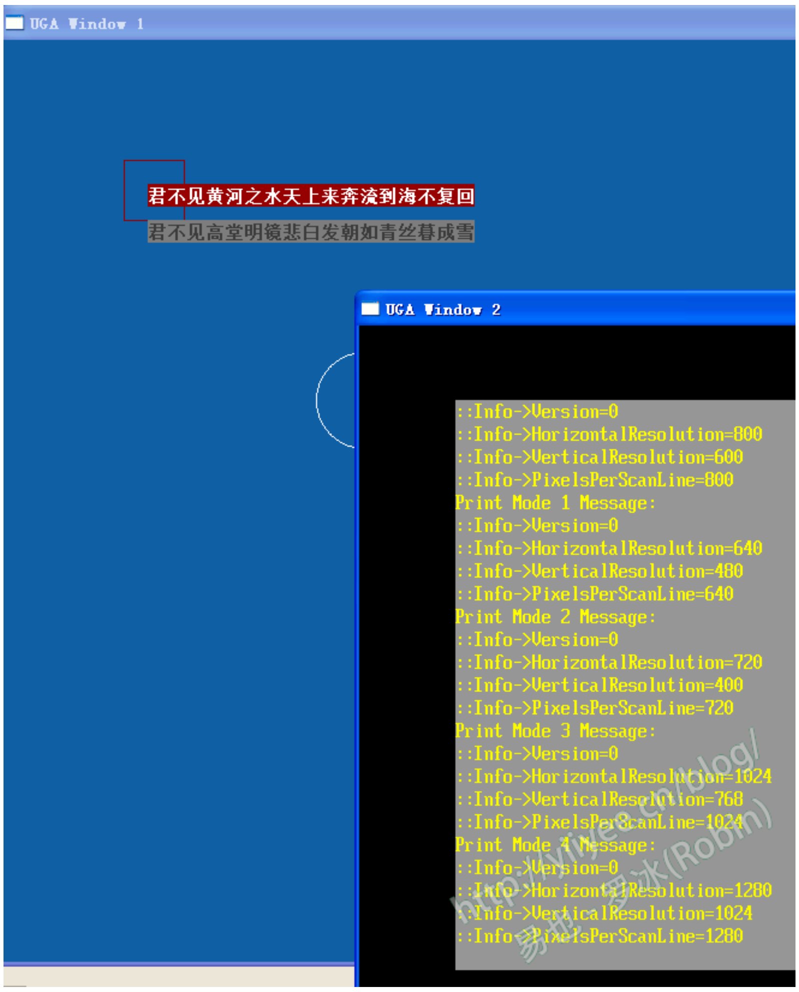
```
MyWorkSpace | Luo3 | Build | IA32
c:\myWorkSpace\_LuoApp\Luo3\*.*
↑文件名
..
Common
Common
example
example
FontData
Graphic
Graphic
HiiFont
HiiFont
Keyboard
Keyboard
LaffStd
Luo3
Luo3
PciRW
PciRW
Window
Window
```

Figure 1 The organized source code

There is not much to explain, mainly some problems encountered during development:

1. The role of FontName. When creating a Font Package or using StringToImage, you will need to set FontName. I just gave it a random name, but I haven't found out what its role is yet. Maybe it will be easier to debug?
2. Variable-length array. When using StringToImage, the CHAR16 FontName[…] member of the EFI_FONT_INFO structure is a variable-length array. This should be a feature supported by C99. I didn't notice it when applying for memory, and directly used sizeof to obtain the length of the structure to apply for it. Obviously, it was not enough, causing the system to crash.
3. In the code of this blog, two fonts are used: SimpleFont and Font. The fonts of SimpleFont exist in example.data, and they are all Unicode Chinese characters. Font also uses this font, and also adds fonts with Ascii codes. Record them down to prevent forgetting later.
4. StringToImage uses several data structures, which can be found in the UEFI Spec. The code I have written so far is only enough for demonstration, and many details have not been considered, including the encapsulation of processing details, interface processing, etc. These codes are not enough for commercial development of Option ROM, so I will organize them when needed.
5. Still StringToImage, the parameter Flags has many values to choose from. I didn't pay attention when debugging, and found that the background color of the text in the reference code could not be changed. I wasted half an hour before I found that the value was incorrect and EFI_HII_OUT_FLAG_TRANSPARENT should be removed. These constant definitions should be carefully checked.
6. There are actually many more effects to explore for text display, such as text scaling, italics, shadows, etc. I think many effects can be achieved by just slightly changing a few parameters.
7. Compared to the dot drawing function I used to display Chinese characters, Hii's method is much more flexible, and the corresponding amount of code is difficult to control. Of course, if you only use English, Hii will be better. In addition, it can also easily switch between languages, which is very suitable for developing a general GUI library.
8. What I like most is that after using Hii, I can directly use L"xxxx" type Unicode strings in the code. Before, when I used the dot drawing method to display Chinese characters, I used the national standard code to find the font, and could not use Unicode strings. This reminded me that I can improve the previous method.
9. The Chinese comma doesn't seem to be extracted. It's a minor problem, so let's leave it alone for now.

Compile the code and execute it as follows:



君不见黄河之水天上来奔流到海不复回
君不见高堂明镜悲白发朝如青丝暮成雪

```
::Info->Version=0
::Info->HorizontalResolution=800
::Info->VerticalResolution=600
::Info->PixelsPerScanLine=800
Print Mode 1 Message:
::Info->Version=0
::Info->HorizontalResolution=640
::Info->VerticalResolution=480
::Info->PixelsPerScanLine=640
Print Mode 2 Message:
::Info->Version=0
::Info->HorizontalResolution=720
::Info->VerticalResolution=400
::Info->PixelsPerScanLine=720
Print Mode 3 Message:
::Info->Version=0
::Info->HorizontalResolution=1024
::Info->VerticalResolution=768
::Info->PixelsPerScanLine=1024
Print Mode 4 Message:
::Info->Version=0
::Info->HorizontalResolution=1280
::Info->VerticalResolution=1024
::Info->PixelsPerScanLine=1280
```

Figure 2 Program running effect

The string defined in the code is as follows:

const CHAR16* TestStr=(const CHAR16 *)L"Have you not seen the Yellow River coming from the sky, rushing to the sea and never coming back?";
const CHAR16* TestStr1=(const CHAR16 *)L"Have you not seen the gray hair in the mirror in the hall, which was black in the morning and turned white in the evening?";

In Figure 2, you can clearly see that the Chinese comma is missing.

So far, we have explored all the contents about Hii multi-text display (mainly for Chinese characters). Although there are still many details to be explored, there are no structural blind spots in understanding.

If UEFI BIOS localization development is needed in the future, these blogs are a good starting point.

Gitee address: https://gitee.com/luobing4365/uefi-explorer
The project code is located under: /12 HiiGraphics-Font.