基于优化存储的嵌入式 GPU 的字符显示①

郭 云, 康 涛, 徐 涵

(特种显示国家工程实验室 现代显示国家重点实验室, 上海 201114)

摘 要: 提出并实现了一种基于嵌入式 GPU(OES: OpenGL® ES)的优化存储的快速字符显示方法. 首先它采用了带宽优化的稀疏阵的存储结构, 它具备良好的空间和时间上的性能优势和可扩展的柔韧性. 同时, 我们采取了静态预定义的字模生成方法, 在选择字符显示时, 通过稀疏存储的索引, 可以快速地定位预定义的字模信息. 我们检查了目前桌面 PC 图形显示环境下(TrueType 等)字符的生成, 索引, 显示的过程, 按调入字库大小, 分别测试字符处理到具体字符显示在画面时所要的时间关系, 明显是逊于本文方法. 同时利用嵌入式 GPU 的多纹理内存的硬件特性, 通过图像预过滤, 实验证明可以保证画面上的字符显示质量.

关键词: 嵌入式 GPU(OpenGL ES); 字符显示; 字符存储; 图形优化技术

Character Display Technology Based on Sparse Storage in the Embedded GPU System

GUO Yun, KANG Tao, XU Han

(National Engineering Laboratory for Special Display, State Key Laboratory of Modern Display, Shanghai 201114, China)

Abstract: A fast character display technology based on the sparse storage is presented. The optimization band storage with sparse structure has the best performance comparing to the previous character storage. And we adopt the predefined glyphs information to index the required character and its appearance data (color etc). Some experiments about computing cost have been finished.

Key words: embedded GPU(OpenGL ES); character display; sparse storage; graphics optimization technigues

1 引言

嵌入式图形显示正在广泛的使用在各种工业和科技领域, 汽车, 自动控制, 移动通信, 航空, 仪器仪表, 游戏娱乐等. 图形显示的内容主要是包括了几何, 图像, 字符等. 由这些内容向用户诠释画面的意义. 字符显示是图形显示中很重要的一个元素, 特别是在各种反映实时状况的显示设备中. 字符显示包括各种文字(各种语言字符, 英文字母, 拉丁字母, 日语片假名, 平假名, 希腊语, 汉字等), 数字, 符号等, 将这些内容根据用户需要反映到一个图形显示画面中, 涉及到诸多的文本设计, 存储, 处理, 存取, 显示的技术. 目前以 ARM, ATI, X86, PPC 为主流的嵌入式设备普遍配置了图形加速引擎, 使用基于 SoC 设计的 GPU(Graphics Process Unit), 这种图形加速硬件可以快速批量的处理数据, 具有较高的图像纹理存储空间, 并且能实现一些特殊的效果(多纹理空间, 透明, 反走样, 雾化, 环境光等), 内置了业界标准的图形加速引擎 OpenGL® ES API. OpenGL® ES 是来源于 PC 机的 OpenGL 1.5, 具有丰富的图形处理能力, 但是精简了许多功能, 包括显示列表 Display List 等, 同时它不提供直接对字符处理的显示支持 API. 在文献[1]中提供了各种 OpenGL 图形环境中的文字显示技术和相应的出处, 并且明确了他们的优缺点. 这里的字符渲染技术主要是使用位图 bitmap, 轮廓 outline(polygon)和纹理 texture map. 字体主要是针对 ASCII 代码, Adobe Type Fonts,还有 FreeType Font[2], 但是字符的处理, 存取, 显示技术普遍依赖于 GLUT 库和显示列表的加速功能. 由于显示列表的功能已经没有被嵌入式图形管线所采纳[3], 而

① 收稿时间:2012-02-29;收到修改稿时间:2012-04-09

且嵌入式领域的 PowerVR[4]等流行的, 非 GLUT 应用 API 的崛起, 促使现有的这些技术很难使用.

同样, FTGL[5]是基于 FreeType2 字库的为桌面 PC 定制的 OpenGL 图形环境里面使用的字符显示 API, 它也沉重的利用了 OpenGL 显示列表和位图中的像素位置定位的功能, 而这些却是嵌入式图形管线所未采用的. FTGL 可以预先调用一个字库进入内存, 在用户选取所需字符时候, 通过所选字符的代码和字模(glyphs)的索引来得到所要的显示字符, 进行实时渲染, 结果传递给图形显示画面.

相对于桌面 PC 的图形环境而言, 嵌入式系统的资源(CPU, GPU 的计算能力, 数据相互调度)比较弱势, 在一个稍微复杂的图形显示画面里面(含有多个不同属性的几何物体, 字符), 要从一个庞大的字符库中检索出用户所需的字符, 执行相应的运算, 这对硬件计算和显示能力是个重要的考验.

我们提出的字符显示是基于 Mark J. Kilgard[6]的字符显示技术, 在这之上, 扩展了字符的存储方法, 特别是汉字的处理, 针对嵌入式系统的图形处理特点, 使用了快速, 高效的检索方法, 在小资源的嵌入式系统环境中可以得到高质量的显示效果.

2 图形系统的字符显示

字符显示是要把一个文字或符号(Character), 通过文本处理程序, 生成一个字模(glyphs), 每个字模通过一个索引映射到文字或符号的代码中. 这些代码根据语言结构不同, 编码也有不同的方法. 中文可以使用固有的区位码. 一个文字或符号所对应的字模可以是一对一的, 也可以是一对多的, 这在微软的 TrueType Open 和 苹果的 TrueType GX 的字模生成中都有不同的处理方法. 字模生成过程中会使用表结构(table)来保存很多的相关信息, 主要包括轮廓点, 度量尺寸, 位图信息, 如下图 1 所示. 其中一个字符或符号在字符串中的位置坐标或对齐方式是很重要的, 这需要文本处理程序通过计算获得, 参看图 2.

字模的上述信息都会存放在表结构中, 主要包括了字模的替换表, 字模的各参数的位置坐标表, 字模的基线偏移表, 字模的间隔调整信息, 字模在字库中的归属信息. 例如替换和位置表中包含有相对于特征列表的偏移信息, 它的表结构是:

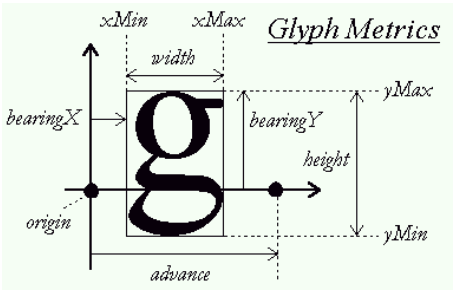


图 1 字符的尺寸参数

图 2 字符在串中的对齐位置

类型	字段名	描述
uint16	FeatureCount	特征记录数
struct	FeatureRecord	特征记录数组
	[FeatureCount]	记录数组的起始索引0 (first feature has FeatureIndex = 0)
		—listed alphabetically by FeatureTag

图 3 字模信息的存储表

文本生成程序会通过描述, 语言, 特征等来检索这些表数据, 传递给字符的渲染程序, 由渲染程序, 产生字符的像素点的信息. 这种像素点的信息可以由图形显示 API 来调用, 显示在图形内容中.

我们检查了上述基于表结构数据存储的字符显示方法并做了测试, 测试的字库来自于微软的 True Type(ttf 类型)字库文件, 参看图 4.

图 4 汉字的显示

对于表结构存储的字符，使用如下的流程执行字模的检索处理:

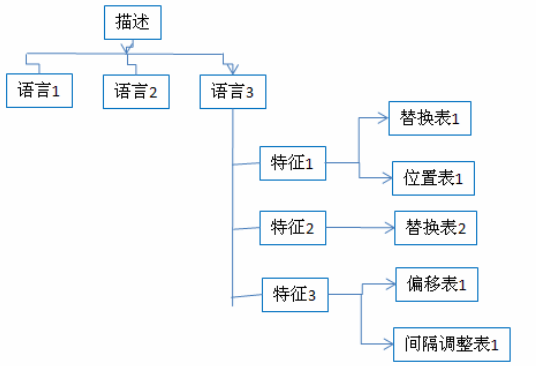


图 5　字符显示程序的表结构检索

执行的程序如下:

```
void FTFont::Render( const wchar_t* string ){
    const wchar_t* c = string;
    pen.X(0); pen.Y(0);
    while( *c){
        if(CheckGlyph( *c)){
            pen = glyphList->Render( *c, *(c + 1), pen);
        }
        ++c;
    }
}
```

这种结构需要我们对用户所调用显示的字符串(string)，做每个字符的检索，由文本程序检索到字模信息，再处理字模的渲染，在一个有 3000-4000 个字符的汉字库中，程序要完成表结构数据存取，计算，字模像素数据的生成等 CPU 的动作，然后系统总线将结果传递给 GPU，使用 OpenGL 图形 API 来完成渲染，得到一帧图像. 我们测试了若干 true type 字库，产生如下关系的图示.

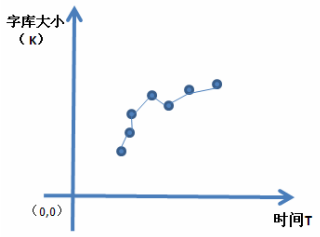


图 6　字库大小和字符显示处理时间

可以看出，表结构的存储所使用的处理时间随预先调用的字库大小，检索的时间代价也是不断上升的. 这对小资源的嵌入式系统的结构和计算资源是个挑战，对于一个复杂画面，势必会影响画面的实时图形显示. 本文开发机(ARM Cortex-A8 和 FreeScale iMX51)中，希望这种检索和字体处理的时间不会有超过线性的费用产生.

3　结构优化的快速字符显示设计

本文设计一个具有柔韧和可伸缩性的数据结构，存储不同的字符和符号，利用 GPU 的图像空间，采用 OpenGL ES 函数调用完成预处理，使用时完成一个简单快速的索引操作，就可以把字符或符号显示在图形内容中. 预处理不会对显示性能产生大的影响. 预处理包括了字体渲染的各种属性数值(颜色，位置等)，剔除了如图 5 所示的各种文本处理的操作. 只需在静态的预存储内存中指定字符位置，快速的把数据传递给 GPU 完成字符的显示，在时间计量上，是一个常量的费用代价. 预处理的字符可以是各种语言和符号的文本. 存储一种语言所需要的符号数据是个很庞大的内存消耗，实际应用中没必要把全部字符数据都调进嵌入式系统. 为此，开发出一个柔韧的，可伸缩的数据存储结构来预存储一定规模的字符或符号，来满足嵌入式应用中快速，高质量字符显示的目标.

首先考察汉字的排列定义,国标(GB 2312-80)中，第一级汉字以拼音字母为序进行排列，同音字以笔形顺序横、竖、撇、捺、折为序，起笔相同的按第二笔，依次类推. 每个汉字对应有 4 个数字组成的区位码，如下图 7 示例:

ai

埃 1603	挨 1604	哎 1605	唉 1606	哀 1607	皑 1608
癌 1609	蔼 1610	矮 1611	艾 1612	碍 1613	爱 1614
隘 1615	捱 6263				

an

鞍 1616	氨 1617	安 1618	俺 1619	按 1620	暗 1621
岸 1622	胺 1623	案 1624	谙 5847	埯 5991	揞 6278
犴 6577	庵 6654	桉 7281	铵 7907	鹌 8038	黯 8786

图 7　汉字的排列

如果我们需要 ai 中的部分汉字，那么就会出现部

分位置的汉字没有进入选择，如图 8.

ai

埃 1603	挨 1604			哀 1607	皑 1608
癌 1609	蔼 1610	矮 1611	艾 1612	碍 1613	
	捱 6263				

图 8 图形显示需要的部分汉字

其中，汉字字符 “艾 1612”是作为可增删的操作结果存放进数据结构中.

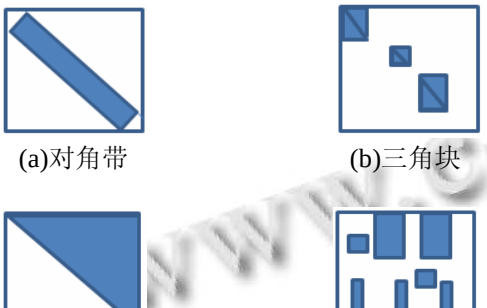
汉字以外的多种语言，它的字符符号也可以使用与汉字类似的两维数据结构来表达他们的定位，日语的 50 音图，就是一个例子:

あ

会う	逢う	あい	遇え	遭え	青
赤	飽き	開く	あけ	亜子	あな
あに	ある				

图 9 日语的 50 音顺的字符排列

类似于图 8 的数据排列存在着稀疏的特点，我们考察工程中对于矩阵表达的大规模线性方程组的求解中，普遍使用了稀疏矩阵的存储结构[7]，它的显著优点是节省时间和空间，如下图 10.



(a)对角带 (b)三角块

(c)有界三角块 (d)双界三角块

图 10 稀疏阵的典型表达

图 10 中的(a)(b)(c)(d)方块大小，是我们预留的二维空间，实际存储时候，那些空白的地方都没有内存消耗. 方块内部有色的小块是我们字符的内存地址，有实际的字符数据. 本文的字符数据采用了这种结构，节约了大量的无为的内存，最大的优点在于它可以建立一个可扩展的柔性字库空间，适用于不同的应用. 同时，我们对该稀疏结构的每个字符单元都建立了地址的静态预定义，因此在调用字符显示时，不需文本处理程序的介入，也不要查找索引表，就可以索引到预先存储好的 OpenGL ES 纹理渲染过的字模信息. 包括了二次的线性过滤信息.

4 结构优化的字符显示的性能

使用上述优化的稀疏阵的字符存储方法，调用 OpenGL ES 的多纹理的缓存处理函数，可以在嵌入式设备上任意操作一个字符的变化，包括移动，旋转，透明，颜色过滤等. 存储空间的优化和柔韧性，让我们的应用程序可以扩展或增删不同文字，符号，建立一个应用所需要的字符库，扩展到图形显示内容中，图 11 是在 FreeScale 的 IMX51 GPU 上，使用 OpenGL ES1.0 的图形 API，利用结构化的字符处理模块，显示的字符和符号,它和 OpenGL ES 的任何一个体素 (GL_POINTS，GL_LINE_STRIP，GL_LINE_LOOP，GL_LINES)一样，可以被做任意的变化和效果处理.

图 11 Freescale iMX51GPU 上的字符显示

5 结论

本文提出并实现了一种基于嵌入式 GPU(OES: OpenGL® ES)的优化存储的快速字符显示方法. 首先它采用了带宽优化的稀疏阵的存储结构，具有很好的

（下转第 158 页）