基于嵌入式设备的动态图形快速绘制方法

丁 鹏[1]
DING Peng

摘 要 随着嵌入式电子设备的发展，嵌入式系统下图形动态显示成为用户的普遍需求，然而为显示更加丰富的图像信息，往往需以消耗更多的内存资源作为代价，如何利用有限的内存来实现图形动态显示是一个关键问题。针对嵌入式设备内存小的特点，研究了在嵌入式设备上如何减小内存使用率的前提下，实现动态图形快速绘制的方法。首先，提出了使用一维数据数组代替原先二维数据数组来减小内存资源的浪费，同时在图形绘制上使用贴图移位来提高绘图速度的方法。然后研究了目前普遍使用的 OpenGL 绘图方法，通过 GPU 来加速绘图效率。最后，两种方法的图形绘制速度比原来方法的图形绘制速度提高 10 倍以上。

关键词 贴图移位；低内存；动态绘制；OpenGl；GPU

doi：10.3969/j.issn.1672-9528.2024.06.047

0 引言

随着电子测量设备的不断发展，三维图形动态显示技术被越来越多用户所使用，三维分别为时间 - 频率 - 幅度，三维绘图显示不仅可以直观地显示信号在频域的强弱分布，同时也可以动态记录一段时间内的信号变化，测试人员无需实时观察，只需要通过分析记录的三维图形获得需要测量的信号特性。为了显示更加丰富的图像信息，往往需要消耗更多的内存资源作为代价，因此如何利用有限的内存来实现三维图形的动态绘制是一个关键问题 [1]。

现有三维图形动态显示技术方案主要采用数据读取模块、颜色数据生成模块、颜色数据缓存模块以及最终的图形绘制模块，数据读取模块实现频谱测量数据的不断更新，颜色数据生成模块通过幅度颜色对照表将数据转换为颜色数据，颜色数据缓存模块按照时间顺序将颜色数据缓存起来，图形绘制模块是将颜色缓存模块中的数据动态地显示在屏幕上。该方法主要是通过遍历颜色缓存模块中的所有二维颜色数组来实现颜色的更新，图形的动态显示是通过移位颜色数组来实现 [2]。图形在绘制时需要不断的去遍历所有的二维颜色数组，由于在图形的动态显示是通过移位二维颜色数组来实现，造成了内存的浪费，而且二维数组在移位交换的过程中降低了程序运行的效率，不利于图形的快速绘制显示。

目前三维图形动态绘制仍然使用 CPU 来绘制，当数据量过大之后，势必造成 CPU 占用的提高，尤其在 CPU 性能不高的嵌入式设备上，可能会引起软件的卡顿，绘图的高 CPU 占用率也会影响软件其它功能的运行。因此在绘图时如何减少 CPU 占用率也是一个需要探索的问题。

综上所述，现有的三维图形动态显示技术需要占用大量的内存和 CPU，降低了程序运行的效率，不利于图形的快速绘制显示。

1 基本原理

为了克服现有技术中存在的缺陷，基于频谱分析仪平台，本文提供一种低内存消耗的三维动态图形绘制方法 [3]。该方法避免对二维数组的初始化分配，节省了内存空间，而且在图形显示时，不需要去遍历数组，加快了画图程序的运行效率，原理如图 1 所示。

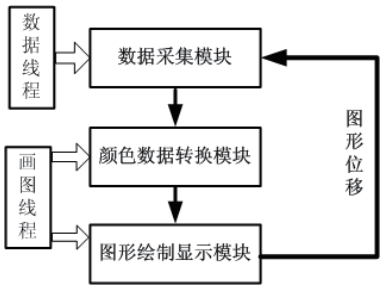

图 1 基本原理框图

具体步骤如下。

步骤一：对频谱分析仪扫描得到的数据进行采集，数据线程完成对频谱扫描测量数据的采集，将采集的数据放在一个一维数组中，所述一维数组的大小与频谱仪扫描点数相等。

步骤二：当完成一次数据采集后，数据数组交由画图线程，画图线程对一维数组中的每个数据点进行幅度—颜色

1. 中电科思仪科技股份有限公司 山东青岛 266555

转换。

步骤三：将经过颜色转换后得到的图形显示在屏幕上。

步骤四：对显示在屏幕上的图形进行向上或者向下的移动，实现动态的图形显示。

步骤五：重复上述步骤一至步骤四的内容，直至整个三维动态图形显示在屏幕上，即可获得三维动态图形。

通过图形位移的方法可以减少内存的占用，但绘图依旧是采用 CPU 来绘制的，而目前流行的 OpenGl 绘图方法 [4]，可以利用 GPU 一次性将数据绘制到屏幕上 , 将大大提高绘图效率，因此本文将探索使用 OpenGl 绘制动态图形的方法及效率。

OpenGl 是一种跨平台的图形库，用于创建二维和三维图形的应用程序，其原理涉及计算机图形学和计算机图形渲染技术。OpenGl 将图形渲染分成两个阶段：几何阶段和光栅化阶段。在几何阶段，OpenGl 会接收应用程序传递来的顶点数据，进行变换、投影和光照的计算，并生成一组裁剪后的顶点。这些顶点最终组成了图形的几何形状。在光栅化阶段，OpenGl 会将裁剪后的几何图形转换成像素，对每个像素进行颜色值的计算和深度值的比较，然后将结果存储到帧缓冲区中，最终显示到屏幕上。在 OpenGl 中，应用程序通过调用 OpenGl API 来传递数据和控制图形渲染流程。OpenGl API 提供了一组函数，可以设置顶点属性、纹理贴图、光照属性等。应用程序可以通过着色器程序来自定义顶点和像素的计算方法，实现复杂的图形渲染效果。

2 实现

软件采用多线程设计 [5]，数据线程与绘图线程配合完成数据的采集与显示，数据线程完成对频谱扫描测量数据的采集，数据采集模块将得到的采集数据存放在一个一维数组中，在数据采集模块完成一次数据采集后，数据数组交由画图线程，画图线程首先对数据数组中的每个数据点进行幅度 - 颜色转换，转换是根据幅度 - 颜色对照表，幅度 - 颜色对照表按照红色为幅度最高，对应的 RGB 值为（255,0,0），蓝色为幅度值最低，对应的 RGB 值为（0,0,255）。幅度值最高与最低之间划分 100 个颜色等级，划分颜色等级的标准是频谱仪显示的信号幅度的变化范围，保证幅度值每 1 dB 的变化都能有不同的颜色对应。数据幅度值经过颜色转换后，则可直接显示在当前屏幕上。

为了实现动态的图形显示，需要将图形向上或者向下移动，可直接使用软件贴图位移的方法，如图 2 变化到图 4，其中 A 点表示显示区域的左上角的坐标，x_1 表示横坐标，y_1 表示纵坐标，H 表示显示区域的高，W 表示显示区域的宽，具体实现方法如下。

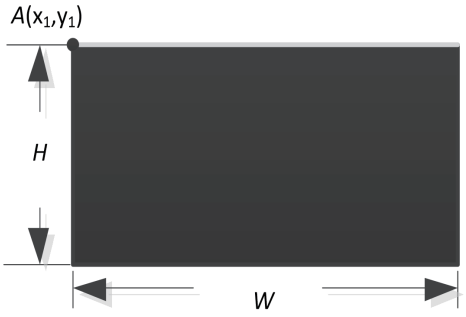

图 2 第一次绘制显示图

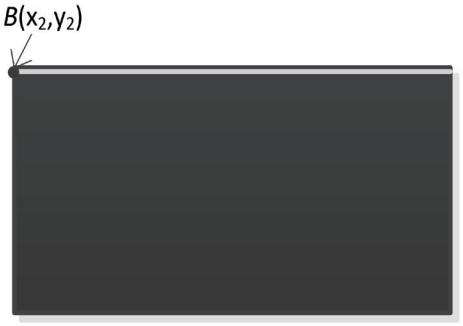

图 3 贴图位移中间过程

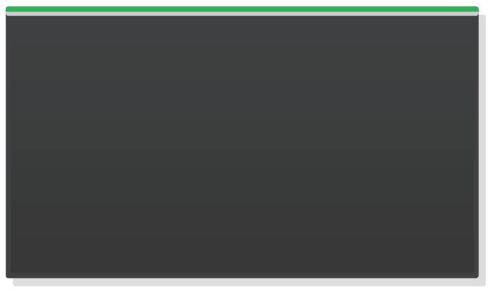

图 4 第二次绘制显示图

此处使用的是 MFC 的绘图框架，首先建立一个内存 DC，定义为 pDC，pDC 用来显示程序中所有画图显示，然后定义另外一个内存 DC，定义为 pCompatibleDC，使其继承于 pDC。

为了实现由图 2 变化到图 4 的过程，具体方法为：将 pDC 内存中的图形拷贝到 pCompatibleDC 中，直接使用 VC 库函数 BitBlt 函数来实现；然后再将 pCompatibleDC 内存中的图形移位拷贝回到 pDC 中，即 BitBlt 时选择左上角的坐标不能再使用 A 点，而需要选用 B 点，B 点的坐标可以设为 $x_2=x_1$，$y_2=y_1+1$；最后再使用 BitBlt 显示到图 4 时，直接将当前颜色转换后的数据显示即可完成整个图形的移动。

由以上实现可见，贴图位移的方法使用一维数组实现了原来二维数组实现的动态图形绘制，节省了内存空间。

通过软件贴图位移的方法，一定程度上提高了绘图效率。但该种方法依然是使用 CPU 来绘制的，在嵌入式系统平台中，

有时 CPU 性能是比较低的，此时使用 CPU 绘图会影响应用程序其它功能的运行，最终导致绘图效率不高。因此本文将探索在嵌入式系统平台使用 GPU 来绘图的方法，采用目前流行的 OpenGl ES[6] 图形库来完成动态图形的绘制。

本文选择使用 Qt 开发框架 [7] 来探索 Opengl 绘图方法及效率，Qt 开发框架对 OpenGl 接口进行了封装，使得在 Qt 环境下调用 OpenGl 的接口更加便捷高效。

Qt 绘图框架有 QWidget 和 Qt Quick 两种，选择使用 Qt Quick 框架，使用 QML 编程语言开发界面。整个绘图实现包含四个模块：OpenGl 帧缓冲模块、OpenGl 渲染模块、OpenGl 数据源模块、OpenGl 绘制模块。几个模块的关系如图 5 所示。

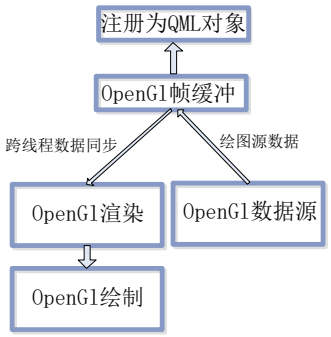

图 5 OpenGl 绘图各模块关系

通过 Qt 元对象系统将 OpenGl 帧缓冲对象注册为 QML 对象，可以实现绘制的界面在 QML 中显示，由此可见 OpenGl 帧缓冲是连接 QML 与 OpenGl 的桥梁。要实现 OpenGl 帧缓冲模块需要继承 Qt 的 QQuickFramebufferObject 类，该类的目的是使用缓冲区对象将 OpenGl 的渲染显示到 Qt Quick 界面。

OpenGl 数据源模块提供绘图的原始数据，在本次实验中存储的是瀑布图数据，数据是一个二维数组，数组中的每个元素是一个颜色值。数据源模块负责不断的获取原始数据，并将原始数据传入 OpenGl 帧缓冲模块。

OpenGl 渲染模块是包含独立的渲染线程，用户实现跨线程的数据同步，实现该模块只需继承 Qt 的 QQuickFramebufferObject::Renderer，该类是 Qt 提供的渲染类。可以重写该类的虚函数实现跨线程的数据同步和数据绘制。在该类中可以直接调用 OpenGl 绘制模块完成绘制。

OpenGl 绘制模块负责具体数据的绘制，该模块通过调用 Qt 封装的 OpenGl 函数接口实现具体单元的绘制，包括加载绘图数据的顶点着色器以及片段着色器，初始化 OpenGl 绘制需要的顶点缓冲区对象和顶点索引对象，设置顶点属性，进行具体的绘制。OpenGl 绘图是利用 GPU 一次性绘制所有像素点的数据，因此需要编写在 GPU 上执行的代码段，即着色器源代码。

顶点着色器代码如下：

```
attribute highp vec2 a_position;
attribute mediump float a_color;
uniform highp mat4 u_mvpMatrix;
uniform mediump float u_redGate;
uniform mediump float u_blueGate;
uniform mediump float u_pointSize;
varying mediump float v_color;
void main(void)
{
gl_PointSize = u_pointSize;
gl_Position = u_mvpMatrix * vec4 (a_position.xy, 0.0, 1.0);
if (a_color <= u_blueGate )
v_color=-1.0;
else if (a_color >= u_redGate)
v_color = 0.999;
else
v_color=(a_color-u_blueGate) / (u_redGate - u_blueGate);
}
```

片段着色器代码如下：

```
precision mediump float;
varying float v_color;
uniform sampler2D s_texture;
void main(void)
{
if (v_color < 0.0)
gl_FragColor = vec4(0.0,0.0,0.0,0.0);
else if (v_color > 0.999);
gl_FragColor=vec4(texture2D(s_texture, vec2(0.999,0.5)).xyz , 1.0);
else
gl_FragColor = vec4(texture2D(s_texture, vec2(v_color,0.5)).xyz , 1.0);
}
```

基于以上各个模块的配合，在数据线程完成对频谱扫描测量数据的采集后，数据线程将采集后的数据更新到一个二维数组中，并将该数组交由画图线程。画图线程中的 OpenGl 数据源模块负责将二维数组中的数据转化为颜色值，并将转换后的数据塞入 OpenGl 帧缓冲模块，然后在 OpenGl 渲染模块进行数据的渲染和绘制，利用着色器代码一次性将二维数组中的每个数据渲染到整个界面中，同时 QML 对象会自动更新显示最新的绘图数据。

3 测试

本方法基于频谱分析仪平台，绘制瀑布图数据，完成图形如图 6 所示。首先我们选用的数据是横轴每条轨迹 1000 点的数据，纵向最多 300 条轨迹，即最大数据量为 300×1000 的数组，数组中每个元素为颜色值。经测试使用贴图移位的方法，实现动态图形绘制每完成一次绘图显示用时在 60 ms 左右，使用 OpenGl 绘图方法实现动态图形绘制每完成一次绘图显示用时在 20 ms 左右，使用原来描点划线的方式完成一次绘图显示用时在 500 ms 左右。

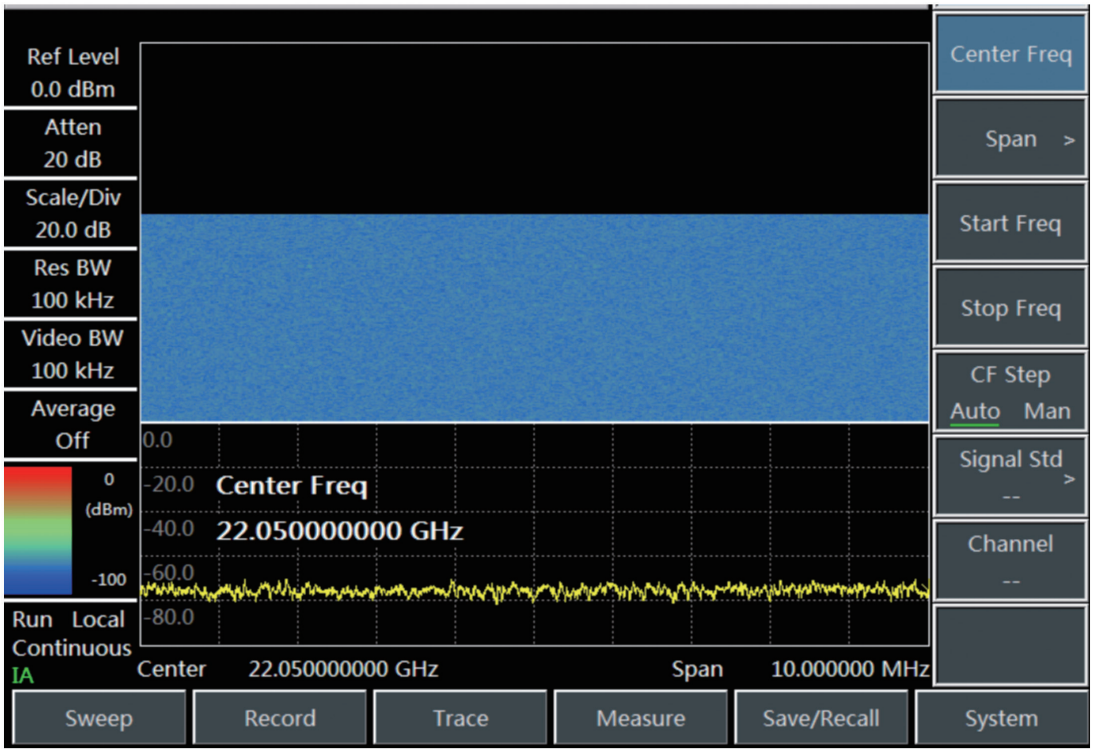

图 6 三维动态图形绘制结果图

然后把横轴每条轨迹的数据点数增加到 4000 点，即最大数据量为 300×4000 时，经测试使用贴图移位的方法实现动态图形绘制每完成一次绘图显示用时在 200 ms 左右，使用 OpenGl 绘图方法实现动态图形绘制每完成一次绘图显示用时仍然在 20 ms 左右。

由此可见，贴图移位的方法节省了内存空间，将绘图效率提升了近 10 倍，但随着绘图数据量的增加，绘图速度会下降。OpenGl 绘图方法节省了 CPU 资源占用，将绘图效率提升了 20 多倍，且在绘图数据量增加后，绘图效率影响不大。

4 结论

本文探索了嵌入式设备上的动态图形快速绘制方法。贴图移位的方法将二维数组替换为一维数组，大大节省了内存空间，绘图效率也提升近 10 倍。OpenGl 绘图方法，使用 GPU 进行图形渲染，减少了对 CPU 资源的占用，绘图效率提升 20 多倍。嵌入式设备在进行动态图形绘制时，可依据需要选择这两种方式进行绘制。若嵌入式设备具备 GPU 模块，则可使用 OpenGl 绘图方式，可大大提高绘图效率。在不具备 GPU 的嵌入式设备上，可使用贴图移位的方法进行绘制，在节省内存空间的同时，也提高了绘图速度。电子测量设备通过使用高效动态图形绘制方法，可快速捕捉突变信号，方便用户更好地观察和分析突变信号的各种信息。

参考文献：

[1] 王文菊 . 瀑布图分析跳频信号软件模块的实现 [J]. 通信对抗 , 2004(2):31-35.

[2] 石俊 . 电磁频谱监测系统中数据采集与实时监控子系统设计与实现 [D]. 西安 : 西安电子科技大学 ,2012.

[3] 赵育才 . 无线电波传播预测与干扰分析研究及实现 [D]. 长沙 : 国防科学技术大学 ,2009.

[4]SELLERS G, WRIGHT R, HAEMEL N. OpenGl 超级宝典 [M]. 北京 : 电子工业出版社 ,2012.

[5]HUGHES C.C++ 面向对象多线程编程 [M]. 北京 : 人民邮电出版社 ,2003.

[6] 甘岚 , 李金标 .OpenGL ES 及 Qt/E 在嵌入式系统中的研究与应用 [J]. 微计算机信息 ,2009,25(35):68-70.

[7]SUMMERFIELD M.Qt 高级编程 [M]. 北京 : 电子工业出版社 , 2018.

【作者简介】

丁鹏（1987—），男，硕士研究生，工程师，研究方向：嵌入式软件开发、频谱分析仪软件开发。

（收稿日期：2024-05-06）