(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 108573524 B
(45)授权公告日 2022.02.08

(21)申请号 201810330487.8

(22)申请日 2018.04.12

(65)同一申请的已公布的文献号
申请公布号 CN 108573524 A

(43)申请公布日 2018.09.25

(73)专利权人 东南大学
地址 210000 江苏省南京市玄武区新街口街道四牌楼2号

(72)发明人 胡轶宁 郑涛 谢理哲 张宇宁 王征

(74)专利代理机构 南京正联知识产权代理有限公司 32243
代理人 王素琴

(51)Int.Cl.
G06T 15/20(2011.01)
G06T 15/04(2011.01)
G06T 15/55(2011.01)

(56)对比文件
CN 101482978 A,2009.07.15
CN 104504671 A,2015.04.08
WO 2017092335 A1,2017.06.08
US 2007103465 A1,2007.05.10
EP 2949121 A1,2015.12.02

审查员 张驰

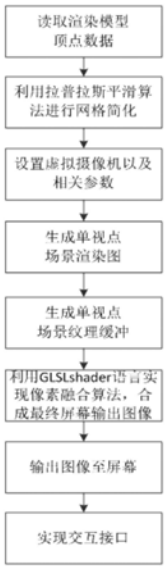
权利要求书3页 说明书7页 附图2页

(54)发明名称

基于渲染管线的交互式实时自由立体显示方法

(57)摘要

本发明提供一种基于渲染管线的交互式实时自由立体显示方法，读入需要渲染的模型的顶点数据，利用顶点数据生成网格模型并得到简化网格模型，传入OpenGL渲染流水线；生成单视点渲染场景图；设置渲染窗口分辨率、目标表面材质、光源类型和位置，利用OpenInventor开源库，分别针对每个不同视点的场景，实施场景渲染；对不同视点场景纹理缓冲进行融合的像素融合算法，得到用于输出的融合图像；利用OpenGL可编程渲染管线特性，使用GLSLshader语言，在片元着色器中完成像素选取与融合操作，最后输出融合后的结果；实现用户交互接口。该方法通过多视角融合实现三维数据的立体显示；利用多纹理映射技术完成每视点处场景的渲染，满足使用者对观察对象进行实时交互式观察的需求。



CN 108573524 B

1.一种基于渲染管线的交互式实时自由立体显示方法,其特征在于:包括以下步骤,

S1、读入需要渲染的模型的顶点数据,利用顶点数据生成网格模型,并利用拉普拉斯平滑算法进行网格简化,得到简化网格模型,传入OpenGL渲染流水线;

S2、根据使用场景设置视点个数、虚拟摄像机位,具体为,设置融合图像的视点个数为N、视点间隔角度delta,调用OpenGL API gluLookAt函数和gluPerspective函数,在以原点为中心点,半径为R的弧上根据视点个数N每相隔角度delta摆放N个虚拟摄像机,且使虚拟摄像机阵列以xoz面对称,其中每个虚拟摄像机的光轴为该位置到原点确定的方向向量,虚拟摄像机的法线方向为坐标轴z的正方向,针对每一个虚拟摄像机,使用OpenGL固定管线渲染步骤S1所述简化网格模型,生成单视点渲染场景图;

S3、设置渲染窗口分辨率、目标表面材质、光源类型和位置,利用OpenInventor开源库,分别针对每个不同视点的场景,实施场景渲染,具体为,调用OpenGL API,使用glGenBuffers、glBindBuffer和glBufferData三个函数在显存中开辟数据空间PixBuffer^{i}即纹理数组,记屏幕横向像素个数为X_w,纵向像素个数为Y_w,则其中每块区域所占大小为$X_w \times Y_w \times 3$比特,将步骤S2所述单视点渲染场景图的帧缓冲区中的颜色缓冲区附加到对应的纹理缓冲区对象,调用glFramebufferTexture2D函数将渲染结果以纹理形式保存到显存中,得到单视点场景纹理缓冲;

S4、将步骤S3所述不同视点场景纹理缓冲进行融合的像素融合算法具体描述如下:遍历屏幕区域所有像素,根据映射公式$N_k = \frac{(i - 3j\tan\alpha) \bmod X}{X} N$,从不同视点处的图像中选取像素通道,其中,i/3的商表示像素横坐标,其范围为0至X_w-1;j表示像素的纵坐标,其范围为0至Y_w-1;i/3的余数为0、1、2时分别对应像素(i/3,j)的蓝、绿、红通道,X为液晶显示屏上单个柱状透镜宽度下所能覆盖的子像素个数,α为倾斜透镜与竖直方向的夹角弧度,N为S2所述视点个数,计算结果N_k为当前子像素所对应的场景纹理编号;根据N_k从对应视点的渲染场景图中获得像素通道值,完成所有像素遍历后即得到用于输出的融合图像;

S5、利用OpenGL可编程渲染管线特性,使用GLSLshader语言,在片元着色器中实现像素融合算法,将步骤S3所述单视点场景纹理缓冲融合生成最终屏幕输出图像,具体为,向片元着色器中传入uniform变量以表示映射公式中的值α和X,根据步骤S4所述映射公式计算出对应坐标处的像素通道值所对应的场景纹理缓冲编号;通过采样器,调用着色器语言内置采样函数texture2D,由计算出的视点索引值选取对应纹理处的颜色值,融合生成片元处的最终颜色值,从而生成具有裸眼3D效果的每一帧图像;

S6、实现用户交互接口,自定义消息回调函数,针对交互时键盘鼠标发出的消息,进行相应的处理,以响应来自用户的交互请求。

2.如权利要求1所述的基于渲染管线的交互式实时自由立体显示方法,其特征在于:步骤S1中,采用拉普拉斯平滑算法得到简化网格模型,具体为:

S11、初始化网格的邻接结构集M;

S12、新建临时点集$\overline{V}$,用来存储集合M中所有点平滑后的位置;

S13、对于所有网格中的顶点V,先初始化临时向量为零向量V_0,接着取的邻域点集P adj(p),再对所有领域点T,将其位置加到临时向量V_t里,最后将临时向量V_t的位置存入临时点

集$\overline{V}$中；

S14、对所有网格中的顶点P，将P的位置修改为临时点集$\overline{V}$中对应的位置。

3.如权利要求1所述的基于渲染管线的交互式实时自由立体显示方法，其特征在于：步骤S2中，视点间隔角度delta根据最佳观看距离dis为参数的公式计算得出，delta＝sin(0.2/dis)。

4.如权利要求1所述的基于渲染管线的交互式实时自由立体显示方法，其特征在于：步骤S4中，访问步骤S3所述单视点场景纹理缓冲的具体方法为：编写片元着色器程序，在着色器中声明sampler2D数组，大小设置为之前的视点个数值N，在着色器程序中通过该数组中的每一个元素访问步骤S3所述场景纹理缓冲，通过gl_FragCoord变量访问片元着色器。

5.如权利要求1-3任一项所述的基于渲染管线的交互式实时自由立体显示方法，其特征在于：步骤S5中裸眼3D效果图像的实时生成具体实施步骤如下：

S51、每一次刷新时将不同视点处的图像采用绘制到纹理技术，此时将帧缓存中的颜色缓冲区RenderBuffer输出到对应纹理对象TextureBuffer上并保存在显存中；

S52、改写渲染管线中的片元着色器，将纹理对象以纹理单元TextureUnit编号，在客户端向着色器传入一个uinform变量，表示纹理对象所在纹理单元，片元着色器根据该句柄访问指定纹理对象；

S53、通过子像素映射公式，通过纹理采样器Sampler以及glsl内置的纹理采样函数texture2d()选取像素并组合成屏幕对应坐标处像素；

S54、首先将屏幕坐标系中的点的二维坐标即着色器语言中的内置变量gl_FragCoord代入以上公式中，从而得出对应于哪一幅图像的索引，之后访问对应视点的图像的纹理对象在对应二维坐标处的颜色信息，最终计算片元的输出值；

S55、建立四边形覆盖整个屏幕，则片元着色器输出的像素点适配整个屏幕，片元着色器此时输出的像素数据即为所需求的数据，且直接通过渲染管线输出。

6.如权利要求1所述的基于渲染管线的交互式实时自由立体显示方法，其特征在于：步骤S6具体为，

S61、首先声明并定义回调函数InventorMotionCallback，作为SoWinExaminerViewer组件的消息回调函数，调用setEventCallback函数完成回调函数在SoWinExaminerViewer类中的注册；

S62、在InventorMotionCallback函数定义中完成其对于Windows窗口消息的处理工作，并针对用户交互时产生的消息刷新场景，以完成对交互的响应。

7.如权利要求6所述的基于渲染管线的交互式实时自由立体显示方法，其特征在于：步骤S62具体为，

S621、当用户按住鼠标左键拖拽时产生鼠标滑动消息WM_MOUSEMOVE，将二维屏幕坐标的变化映射到三维坐标系中，实现虚拟trackball，完成对于每个子场景中物体的旋转、移动变换，同时将变换后的每个子场景重新渲染到对应的纹理对象中，调用重绘函数，渲染管线中的片元着色器针对新的纹理对象执行像素选取融合算法，生成新的输出数据，完成交互后结果的刷新显示；

S622、当用户转动鼠标中间的滚轮时，会产生鼠标滚轮移动消息WM_MOUSEWHEEL，将鼠标滚轮正向与逆向转动角度映射到三维坐标系中，产生场景中物体沿着Z轴方向的平移，同

时将变换后的每个子场景重新渲染到对应的纹理对象中，调用重绘函数，渲染管线中的片元着色器针对新的纹理对象执行像素选取融合算法，生成新的输出数据，完成交互后结果的刷新显示，完成了场景中物体的缩放；

S623、当用户按下键盘相应按键时，产生对应的按键消息，针对不同的按键消息，完成参数的实时调节与场景的重绘。

基于渲染管线的交互式实时自由立体显示方法

技术领域

[0001] 本发明涉及一种基于渲染管线的交互式实时自由立体显示方法。

背景技术

[0002] 相比于传统的二维平面显示,立体显示能够提供给观看者更接近于真实世 界的视觉感受,而自由立体显示技术利用了人眼的视差原理,不通过任何工具 就能让左右眼睛从视屏幕上看到两幅具有视差的,有所区别的画面,将它们反 射到大脑,人就会产生立体感。立体显示技术摆脱了助视设备的束缚,成为了 当前立体显示领域的研究热点。随着立体显示技术的不断发展,已经有多种技 术可用于实现立体显示。

[0003] 基于柱状透镜的自由立体显示技术需要从大量视角对同一场景进行绘制, 对于渲染速度以及大数据量分析与处理速度有很高的要求。目前世面上的自由 立体显示系统往往无法做到根据用户的交互指令,进行实时刷新,采用离线制 作好的视频素材;也就是预先处理每帧图像,将选取像素并融合的多帧图像最 终合成视频,最终在自由立体显示设备上播放。这样的方案存在操作繁琐,播 放内容不能实时更改等局限。

发明内容

[0004] 本发明的目的是提供一种基于渲染管线的交互式实时自由立体显示方法, 通过将传入的数据进行绘制成像,通过倾斜式柱状透镜自由立体显示设备实现 裸眼3D显示效果,基于OpenGL可编程渲染管线开发,实现了实时交互式自 由立体显示,根据用户的交互指令对目标数据进行平移、旋转、缩放等操作, 并可以做到实时刷新绘制,解决现有技术中存在的操作繁琐,播放内容不能实 时更改等局限的问题。

[0005] 本发明的技术解决方案是:

[0006] 一种基于渲染管线的交互式实时自由立体显示方法,包括以下步骤,

[0007] S1、读入需要渲染的模型的顶点数据,利用顶点数据生成网格模型,并利 用拉普拉斯平滑算法进行网格简化,得到简化网格模型,传入OpenGL渲染流 水线;

[0008] S2、根据使用场景设置视点个数、虚拟摄像机位,具体为,设置融合图像 的视点个数为N、视点间隔角度delta,调用OpenGL API gluLookAt函数和gluPerspective函数,在以原点为中心点,半径为R的弧上根据视点个数N每相 隔角度delta摆放N个虚拟摄像机,且使虚拟摄像机阵列以xoz面对称,其中每 个虚拟摄像机的光轴为该位置到原点确定的方向向量,虚拟摄像机的法线方向 为坐标轴z的正方向,针对每一个虚拟摄像机,使用OpenGL固定管线渲染步 骤S1所述简化网格模型,生成单视点渲染场景图。

[0009] S3、设置渲染窗口分辨率、目标表面材质、光源类型和位置,利用 OpenInventor开源库,分别针对每个不同视点的场景,实施场景渲染,具体为, 调用OpenGL API,使用glGenBuffers、glBindBuffer和glBufferData三个函数在 显存中开辟数据空间PixBufferi即纹理数组,记屏幕横向像素个数为X_w,纵向 像素个数为Y_w,则其中每块区域所占大小为$X_w \times Y_w \times 3$比特,,利用Render To Texture技术,将之前使用固定管线生成的单角

度渲染场景图的帧缓冲区中的颜 色缓冲区附加到对应的纹理缓冲区对象，调用glFramebufferTexture2D函数将渲 染结果以纹理形式保存到显存中；

[0010] S4、将步骤S3所述不同视点场景纹理缓冲进行融合的像素融合算法具体描 述如下：遍历屏幕区域所有像素，根据映射公式$N_k=\frac{(i-3j\tan\alpha)\bmod X}{X}N$ ，从 不同视点处的图像中选取像素通道，其中，i/3的商表示像素横坐标，其范围为 0至X_w-1；j表示像素的纵坐标，其范围为0至Y_w-1；i/3的余数为0、1、2 时分别对应像素(i/3，j)的蓝、绿、红通道，X为液晶显示屏上单个柱状透镜 宽度下所能覆盖的子像素个数，α为倾斜透镜与竖直方向的夹角弧度，N为S2 所述视点个数，计算结果N_k为当前子像素所对应的场景纹理编号；根据N_k从 对应视点的渲染场景图中获得像素通道值，完成所有像素遍历后即得到用于输 出的融合图像；

[0011] S5、利用OpenGL可编程渲染管线特性，使用GLSLshader语言，在片元着 色器中实现像素融合算法，将S3所述单视点场景纹理缓冲融合生成最终屏幕输 出图像，，具体为，向片元着色器中传入uniform变量以表示映射公式中的值α 和X，根据S4所述映射公式计算出对应坐标处的像素通道值所对应的场景纹理 缓冲编号；通过该采样器，调用着色器语言内置采样函数texture2D，由计算出 的视点索引值选取对应纹理处的颜色值，融合生成片元处的最终颜色值，从而 生成具有裸眼3D效果的每一帧图像；

[0012] S6、实现用户交互接口，自定义消息回调函数，针对交互时键盘鼠标发出 的消息，进行相应的处理，以响应来自用户的交互请求。

[0013] 进一步地，步骤S1中，采用拉普拉斯平滑算法实现简化得到的网格结构的平 滑操作，具体为：

[0014] S11、初始化网格的邻接结构集M；

[0015] S12、新建临时点集$\overline{V}$，用来存储集合M中所有点平滑后的位置；

[0016] S13、对于所有网格中的顶点V，先初始化临时向量为零向量V_0，接着取 的邻域点集P adj(p)，再对所有领域点T，将其位置加到临时向量V_t里，最后 将临时向量V_t的位置存入临时点集$\overline{V}$中；

[0017] S14、对所有网格中的顶点P，将P的位置修改为临时点集$\overline{V}$ 中对应的位置。

[0018] 进一步地，步骤S2中，视点间隔角度delta是根据最佳观看距离dis为参数 的公式计算得出，delta＝sin(0.2/dis)米。

[0019] 进一步地，步骤S4中，访问步骤S3所述单视点场景纹理缓冲的具体方法 为：编写片元着色器程序，在着色器中声明sampler2D数组，大小设置为之前 的视点个数值N，在着色器程序中通过该数组中的每一个元素访问步骤S3所述 场景纹理缓冲，通过gl_FragCoord变量访问片元着色器。

[0020] 进一步地，步骤S5中裸眼3D效果图像的实时生成具体实施步骤如下：

[0021] S51、每一次刷新时将不同视点处的图像采用绘制到纹理技术，此时将帧缓 存中的颜色缓冲区RenderBuffer输出到对应纹理对象TextureBuffer上并保存在 显存中；

[0022] S52、改写渲染管线中的片元着色器，将纹理对象以纹理单元TextureUnit 编号，在客户端向着色器传入一个uinform变量，表示纹理对象所在纹理单元， 片元着色器根据

该句柄访问指定纹理对象；

[0023] S53、通过子像素映射公式，通过纹理采样器Sampler以及glsl内置的纹理 采样函数texture2d()选取像素并组合成屏幕对应坐标处像素；

[0024] S54、首先将屏幕坐标系中的点的二维坐标即着色器语言中的内置变量 gl_FragCoord代入以上公式中，从而得出对应于哪一幅图像的索引，之后访问 对应视点的图像的纹理对象在对应二维坐标处的颜色信息，最终计算片元的输 出值；

[0025] S55、建立四边形覆盖整个屏幕，则片元着色器输出的像素点适配整个屏幕，片元着色器此时输出的像素数据即为所需求的数据，且直接通过渲染管线输出。

[0026] 进一步地，步骤S6具体为，

[0027] S61、首先声明并定义回调函数InventorMotionCallback，作为SoWinExaminerViewer组件的消息回调函数，调用setEventCallback函数完成回 调函数在SoWinExaminerViewer类中的注册。

[0028] S62、在InventorMotionCallback函数定义中完成其对于Windows窗口消息 的处理工作，并针对用户交互时产生的消息刷新场景，以完成对交互的响应。

[0029] 进一步地，步骤S62具体为，

[0030] S621、当用户按住鼠标左键拖拽时产生鼠标滑动消息WM_MOUSEMOVE，将二维屏幕坐标的变化映射到三维坐标系中，实现虚拟trackball，完成对于每 个子场景中物体的旋转、移动变换，同时将变换后的每个子场景重新渲染到对 应的纹理对象中，调用重绘函数，渲染管线中的片元着色器针对新的纹理对象 执行像素选取融合算法，生成新的输出数据，完成交互后结果的刷新显示；

[0031] S622、当用户转动鼠标中间的滚轮时，会产生鼠标滚轮移动消息 WM_MOUSEWHEEL，将鼠标滚轮正向与逆向转动角度映射到三维坐标系中，产生场景中物体沿着Z轴方向的平移，同时将变换后的每个子场景重新渲染到 对应的纹理对象中，调用重绘函数，渲染管线中的片元着色器针对新的纹理对 象执行像素选取融合算法，生成新的输出数据，完成交互后结果的刷新显示，完成了场景中物体的缩放；

[0032] S623、当用户按下键盘相应按键时，产生对应的按键消息，针对不同的按 键消息，完成参数的实时调节与场景的重绘。

[0033] 本发明的有益效果是：该种基于渲染管线的交互式实时自由立体显示方法，基于可编程渲染管线技术，针对倾斜柱状透镜自由立体显示系统开发，通过多 视角融合实现三维数据的立体显示；利用多纹理映射技术完成每视点处场景的 渲染，满足使用者对观察对象进行实时交互式观察的需求。

附图说明

[0034] 图1是本发明实施例基于渲染管线的交互式实时自由立体显示方法的流程 示意图；

[0035] 图2是本发明实施例中柱透镜单元的光传输特性的示意图。

[0036] 图3是实施例中倾斜透镜自由立体显示屏中视点分布示意图。

[0037] 图4是实施例中视点像素映射关系示意图。

具体实施方式

[0038] 下面结合附图详细说明本发明的优选实施例。

[0039] 实施例

[0040] 实施例的一种基于渲染管线的交互式实时自由立体显示方法，读入需要渲 染的模型的顶点数据，将顶点数据以顶点数组的形式传入OpenGL渲染流水线；根据使用场景设置视点个数、虚拟摄像机位；设置渲染窗口分辨率、目标表面 材质、光源类型和位置，利用OpenInventor开源库，分别针对每个不同视点的 场景，实施场景渲染；利用绘制到纹理显存接口，将场景渲染输出至纹理显存； 利用OpenGL可编程渲染管线特性，使用GLSLshader语言，在片元着色器中完 成像素选取与融合操作，最后输出融合后的结果；实现用户交互接口。该方法 通过多视角融合实现三维数据的立体显示；利用多纹理映射技术完成每视点处 场景的渲染，满足使用者对观察对象进行实时交互式观察的需求。

[0041] 一种基于渲染管线的交互式实时自由立体显示方法，如图1，包括以下步 骤：

[0042] S1、读入需要渲染的模型的顶点数据，利用顶点数据生成网格模型，并利 用拉普拉斯平滑算法进行网格简化，得到简化网格模型，传入OpenGL渲染流 水线。

[0043] 步骤S1中，采用拉普拉斯平滑算法实现了简化得到的网格结构的平滑操作，具体为：

[0044] S11、初始化网格的邻接结构集M；

[0045] S12、新建临时点集$\overline{V}$，用来存储集合M中所有点平滑后的位置；

[0046] S13、对于所有网格中的顶点V，先初始化临时向量为零向量V_0，接着取 的邻域点集P adj(p)，再对所有领域点T，将其位置加到临时向量V_t里(临时向 量/＝领域点集数)，最后将临时向量V_t的位置存入临时点集$\overline{V}$中；

[0047] S14、对所有网格中的顶点P，将P的位置修改为临时点集$\overline{V}$中对应的位置。

[0048] S2、根据使用场景设置视点个数、虚拟摄像机位，具体为，设置融合图像 的视点个数为N、视点间隔角度delta，调用OpenGL API gluLookAt函数和 gluPerspective函数，在以原点为中心点，半径为R的弧上根据视点个数N每相 隔角度delta摆放N个虚拟摄像机，且使虚拟摄像机阵列以xoz面对称，其中每 个虚拟摄像机的光轴为该位置到原点确定的方向向量，虚拟摄像机的法线方向 为坐标轴z的正方向，针对每一个虚拟摄像机，使用OpenGL固定管线渲染步 骤S1所述简化网格模型，生成单视点渲染场景图。

[0049] 步骤S2中，视点间隔角度delta是根据最佳观看距离dis为参数的公式计算 得出，delta＝sin(0.2/dis)，其中dis的单位为米。

[0050] S3、调用OpenGL API，使用glGenBuffers、glBindBuffer和glBufferData三 个函数在显存中开辟数据空间PixBufferi(纹理数组)，记屏幕横向像素个数为 X_w，纵向像素个数为Y_w，则其中每块区域所占大小为$X_w \times Y_w \times 3$比特，利用 Render To Texture技术，将之前使用固定管线生成的单角度渲染场景图的帧缓 冲区中的颜色缓冲区附加到对应的纹理缓冲区对象，调用 glFramebufferTexture2D将渲染结果以纹理形式保存到显存中，同时需要注意的 是每一次操作的时候需要指定一个纹理单元，具体说来是调用glActiveTexture，这样便于标记对应视角渲染结果的纹理，后续着色器中可以访问到。

[0051] 本实施例使用了光栅式自由立体显示技术来实现医疗数据的自由立体显示，该

技术通过光栅的分光特性将不同视差图像在空间上进行分离从而让观看者感 受到立体感。如图2所示为柱透镜单元的光传输特性示意图,F和F’为柱透镜 单元的第一焦点和第二焦点,设透镜焦距为f。设柱透镜后方与透镜距离为g， 与光轴距离为h的一点$_A$经透镜成像于点$_B$。点$_B$与透镜凸面顶点距离为l， 与光轴距离为w,根据几何关系可得：$\frac{h}{f}=\frac{w}{l-f}$，则透镜的横向放大率为：$m=\frac{w}{h}=\frac{l-f}{f}$。

[0052] 本实施例中的自由立体显示器由上述公式建立了物点距离光轴的高度h与 眼睛位置(l,w)的关系。使显示器中的子像素经过其对应的透镜后成像于观看位 置处的相同区域处(该视点图像的最佳观看位置)。本实施例中的倾斜柱状透镜 自由立体显示器使得人眼在最佳观看位置处便能看到立体显示效果,如图3所 示。

[0053] 具体实现步骤说明如下：

[0054] S31、定间隔角度和总视点数,依次渲染生成对应视点处的图像；

[0055] S32、根据像素所在坐标,将相应参数代入像素映射公式中,计算结果N_k， 从第N_k个视点的图像中的选取对应坐标处的像素；

[0056] S33、将选取的所有像素组成的图像放入屏幕所在缓冲区中。

[0057] S4、将步骤S3所述不同视点场景纹理缓冲进行融合的像素融合算法具体描 述如下:遍历屏幕区域所有像素,根据映射公式$N_k=\frac{(i-3j\tan\alpha)\bmod X}{X}N$，从 不同视点处的图像中选取像素通道,其中,i/3的商表示像素横坐标,其范围为 0至X_w-1;j表示像素的纵坐标,其范围为0至Y_w-1;i/3的余数为0、1、2 时分别对应像素(i/3,j)的蓝、绿、红通道,X为液晶显示屏上单个柱状透镜 宽度下所能覆盖的子像素个数,α为倾斜透镜与竖直方向的夹角弧度,N为S2 所述视点个数,计算结果N_k为当前子像素所对应的场景纹理编号;根据N_k从 对应视点的渲染场景图中获得像素通道值,完成所有像素遍历后即得到用于输 出的融合图像。

[0058] 该实例中访问步骤S3所述单视点场景纹理缓冲的具体方法为:编写片元着 色器程序,在着色器中声明sampler2D数组,大小设置为之前的视点个数值N， 在着色器程序中通过该数组中的每一个元素访问步骤S3所述场景纹理缓冲,通 过gl_FragCoord变量访问片元着色器。

[0059] 以x_{off}表示RGB子像素(x,y)距离透镜边缘的水平距离。图4是根据映射公 式计算得出的9视点显示器子像素映射表示意图。

[0060] S5、利用OpenGL可编程渲染管线特性,使用GLSLshader语言,在片元着 色器中实现像素融合算法,将S3所述单视点场景纹理缓冲融合生成最终屏幕输 出图像,具体为:向片元着色器中传入uniform变量以表示映射公式中的值α 和X,根据S4所述映射公式计算出对应坐标处的像素通道值所对应的场景纹理 缓冲编号通过该采样器,调用着色器语言内置采样函数texture2D,由计算出的 视点索引值选取对应纹理处的颜色值,融合生成片元处的最终颜色值,从而生 成具有裸眼3D效果的每一帧图像。

[0061] 该实施例中的裸眼3D效果图像的实时生成具体实施步骤如下：

[0062] S51、每一次刷新时将不同视点处的图像采用绘制到纹理(Render to Texture；RTT)技术，此时将帧缓存中的颜色缓冲区RenderBuffer输出到对应纹 理对象TextureBuffer上并保存在显存中；

[0063] S52、改写渲染管线中的片元着色器，将纹理对象以纹理单元TextureUnit 编号，在客户端向着色器传入一个uinform变量(整形数组)，表示纹理对象所在 纹理单元，片元着色器中就可以根据该句柄访问指定纹理对象；

[0064] S53、通过上文所述的子像素映射公式，通过纹理采样器Sampler以及glsl 内置的纹理采样函数texture2d()选取像素并组合成屏幕对应坐标处像素；

[0065] S54、首先将屏幕坐标系中的点的二维坐标(着色器语言中的内置变量 gl_FragCoord)代入以上公式中，从而得出对应于哪一幅图像的索引，之后访问 对应视点的图像的纹理对象在对应二维坐标处的颜色信息，最终计算片元的输 出值；

[0066] S55、建立四边形覆盖整个屏幕，则片元着色器输出的像素点就可以适配整 个屏幕。片元着色器此时输出的像素数据即为所需求的数据，且直接通过渲染 管线输出，不占用显存与内存额为的带宽，满足实时性需求。

[0067] S6、自定义消息回调函数，针对交互时键盘鼠标发出的消息，进行相应的 处理，以响应来自用户的交互请求。比如旋转、平移、放大、缩小等等。具体 如何完成用户交互功能步骤如下：

[0068] S61、首先声明并定义回调函数InventorMotionCallback，作为SoWin

[0069] ExaminerViewer组件的消息回调函数，调用setEventCallback函数完成回调 函数在SoWinExaminerViewer类中的注册。

[0070] S62、在InventorMotionCallback函数定义中完成其对于Windows窗口消息 的处理工作，并针对用户交互时产生的消息刷新场景，以完成对交互的响应。具体的交互操作及其响应下面具体说明：

[0071] S63、当用户按住鼠标左键拖拽时产生鼠标滑动消息WM_MOUSEMOVE，这时候跳转到相应处理模块，将二维屏幕坐标的变化映射到三维坐标系中，实 现了虚拟trackball，完成了对于每个子场景中物体的旋转，移动变换，同时将 变换后的每个子场景重新渲染到对应的纹理对象中，调用重绘函数，渲染管线 中的片元着色器针对新的纹理对象执行像素选取融合算法，生成新的输出数据， 完成交互后结果的刷新显示。

[0072] S64、当用户转动鼠标中间的滚轮时，会产生鼠标滚轮移动消息 WM_MOUSEWHEEL，这时候跳转到相应处理模块。将鼠标滚轮正向与逆向转 动角度映射到三维坐标系中，产生场景中物体沿着Z轴方向的平移，同时将变 换后的每个子场景重新渲染到对应的纹理对象中，调用重绘函数，渲染管线中 的片元着色器针对新的纹理对象执行像素选取融合算法，生成新的输出数据， 完成交互后结果的刷新显示，完成了场景中物体的缩放。

[0073] S65、当用户按下键盘相应按键时，产生对应的按键消息，这时候跳转到相 应处理模块。针对不同的按键消息，完成参数的实时调节与场景的重绘。

[0074] 实施例通过给观看者左右两眼分别送去不同的画面，从而达到立体的视觉 效果。实施例利用渲染到纹理技术，将多视点场景目标输入到纹理显存；利用 OpenGL可编程管线特性，将多视点像素选取融合操作在流水线中实现，以达 到实时渲染的目的。实施例采取修改OpenGL可编程渲染管线，将相关算法加 入着色器中实时执行的方法来优化数据通路，

充分利用了显卡运算硬件上本身 具有的高并发特性,而且数据的处理全过程都在显卡端完成,没有了数据传输 的时延,因此可以做到实时交互。

[0075] 该种基于渲染管线的交互式实时自由立体显示方法,基于可编程渲染管线 技术,针对倾斜柱状透镜自由立体显示实施例开发,通过多视角融合实现三维 数据的立体显示;利用多纹理映射技术完成每视点处场景的渲染,满足使用者 对观察对象进行实时交互式观察的需求。

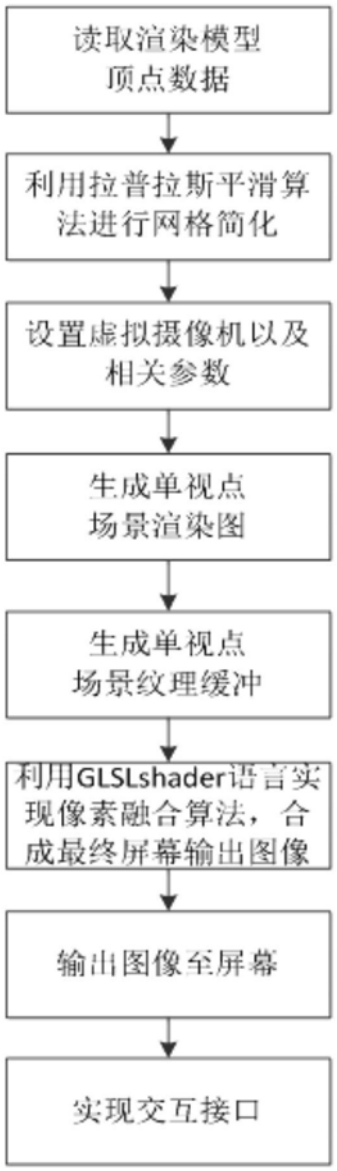

读取渲染模型
顶点数据
利用拉普拉斯平滑算
法进行网格简化
设置虚拟摄像机以及
相关参数
生成单视点
场景渲染图
生成单视点
场景纹理缓冲
利用GLSLshader语言实
现像素融合算法，合
成最终屏幕输出图像
输出图像至屏幕
实现交互接口

图1

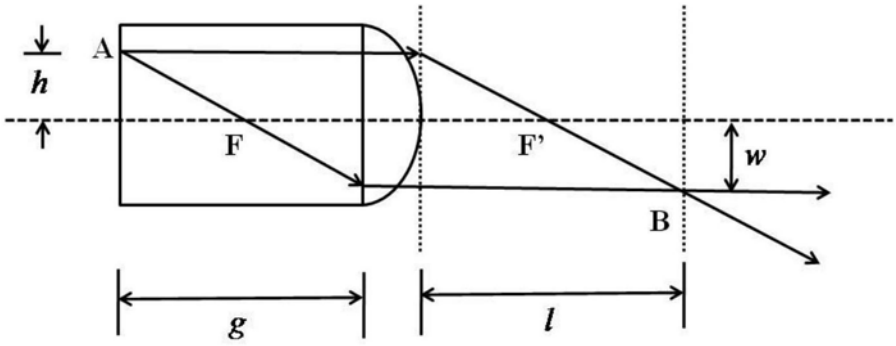

A
h
F
F'
w
B
g
l

图2

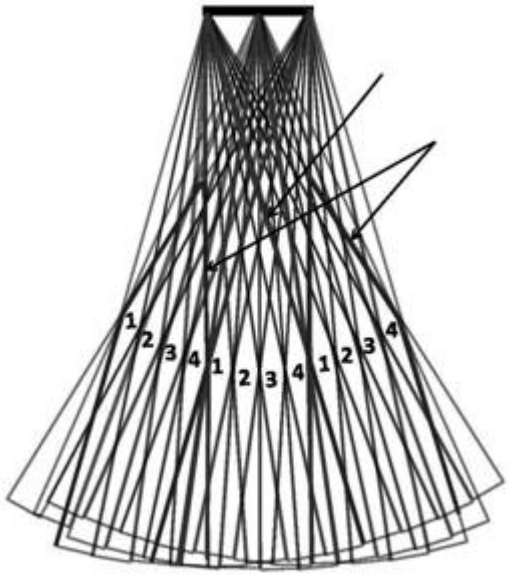


图3

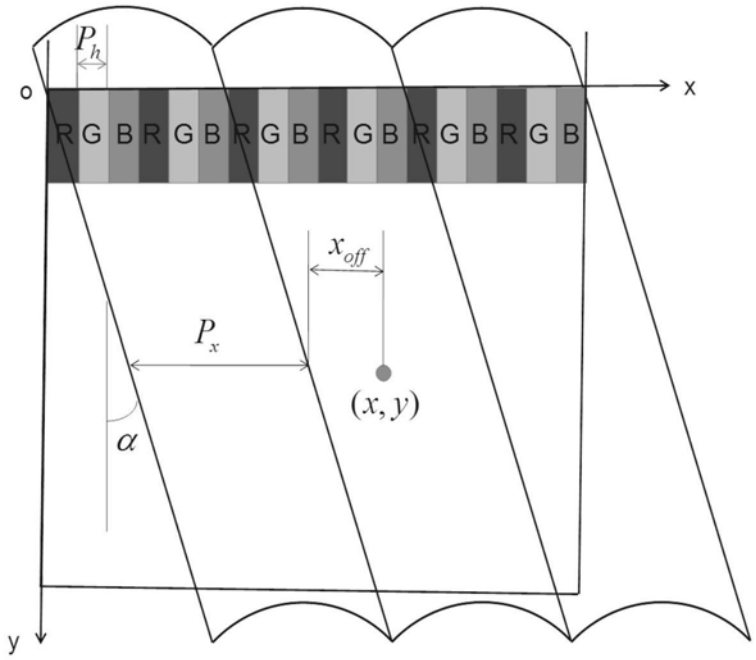


图4