支持 GPU 的 Flash3D 技术①

钱 蔚

(国网电力科学研究院, 南京 210000)

摘 要: 3D 图形技术的应用早已十分广泛. 随着互联网的发展, 越来越多的应用被推广到互联网上, 但涉及 3D 的应用却不多, 近年来, 多种新技术的发展给基于 Web 的 3D 图形技术注入了新的活力. 本文主要研究了基于 Adobe Flash 平台的 Web3D 技术. 首先介绍了 Flash3D 的工作原理, 然后介绍了相关的一些 3D 引擎, 最后基于其中的一个 3D 引擎研究了创建 3D 场景的一系列关键技术, 并实现了一个演示范例.

关键词: Flash3D; Stage3D; Flare3D; 碰撞检测

Flash3D Technology Which Support GPU

QIAN Wei

(State Grid Electric Power Research Institute, Nanjing 210000, China)

Abstract: Application of 3D is already a wide range used technology. With the development of Internet, more and more applications are extended to the Internet, but the application relates to the 3D is not enough, in recent years, the development of many new technologies for 3D technology based on Web has injected new vitality. This paper mainly studies Web3D Technology based on the Adobe Flash platform. First introduces the work principle of Flash3D, then describes some 3D engine, finally, based on a 3D engine, the paper researched a series of key technologies to create a 3D scene, and realize a demo example.

Key words: Flash3D; Stage3D; Flare3D; collision detection

Flash3D 是指所有基于 Adobe® Flash®平台(基于网页的 Flash Player 或基于桌面的 AIR)播放的可交互的实时三维应用的总称. Flash Player 10 中已经内置了 2.5D 绘图支持、3D 属性、绘制空间三角形的 API、允许处理透视等特性[1-3]. 但这些利用软件渲染的 3D 特效无法在复杂度极高的场景中应用. 究其原因, Flash Player 10 平台还不支持 GPU, 而只能基于 CPU 运算, 所以表现复杂图像时往往显得力不从心. 基于 Flash Player 10 的 3d 引擎仅能实时渲染数千个面.

2011 年 10 月, Adobe 公司发布了新的 Flash 平台(Flash Player 11 和 Adobe AIR 3). 该平台提供了一个新的硬件加速 3D 渲染引擎 Stage 3D, Stage3D 所包含的底层 API 会让 3D 应用充分利用 GPU 性能, 从而得到惊人的效率提升. 可以在 60Hz 左右的高分辨率下渲染近百万的 Z-Buffer 三角形.

Stage3D 应用可以在众多平台上运行. 用户可以通过 Flash 运行时的跨平台特性而在各种支持 Flash 的设备上体验到增强的 3D 感受. 你可以将 Stage3D 应用编译成 AIR 应用也可以用 Flash Player 来播放它们. 这意味着你可以使用 Stage3D 创建桌面 3D 或 Web3D 应用. 同时, 也有可能以同样的代码将应用配置到移动平台, 例如 iOS 和 Android. 借助于安装了 Flash Player 巨大数量的群体, Stage3D 将会有广泛的应用范围.

1 Stage3D简介

Stage3D 是一组可编程的基于 Shader 的 3D API(Shader 称为渲染或着色器, 是一段对 3D 对象进行操作、并可以由 GPU 所执行的程序). Stage3D 允许开发者在可能的情况下调用 GPU, 同时也提供当 GPU, 不兼容的情况下使用 CPU 来做备用处理器的方案. 在

① 收稿时间:2013-06-20;收到修改稿时间:2013-07-15

Windows 系统中将会依赖于 DirectX 9, 在 Mac 和 Linux 下则依赖于 OpenGL 1.3, 在 Android 和 Linux Mobile 等移动平台则借助 OpenGL ES 2.0 提供支持.

使用 DirectX 和 OpenGL 这种本地 APIs. 为了能挖掘出显卡的全部功能以及特效, 必须提供完整的硬件信息. 为了发挥出特定 GPU 的性能. 还经常有必要调整代码, 使其能充分利用到硬件的特性. 这样开发者会在最先进的硬件上创建最震撼的效果. 但是这同样意味着应用必须要针对不同的硬件做调整和测试.

使用 Stage3D. 你只需按照 Stage3D 的概念编程, 你建立的应用就可以在每个支持 Flash Player 或 AIR 的平台上运行. Stage3D 就像一个介于你的代码和真实的硬件之间的中间层. 所以, Stage3D 更易用. 当然, 这也有一些缺点, 因为 Stage3D 不是为了发挥每个平台特有的能力而专门针对某个硬件开发的.

2 Stage3D工作原理

Flash 是基于舞台(Stage)的概念设计的. 每一个在 Flash 中显示的物体都是一个添加到舞台上的 DisplayObject. 所以, 舞台是一个盛放所有被显示事物的容器. 它是所有 2D 事项的根. 当 Adobe 在 Flash 中引入 3D 渲染时, 他们加入了一系列新的专门针对于 3D 的特殊舞台. 这些特殊舞台被称为 Stage3D. 它们被定位在主 Flash 舞台后面. Stage3D 创建的 3D 内容在每一个具体的 Stage3D 的矩形视口中被渲染, 然后正常的 2D Flash 内容覆盖在它们上面. 如图 1 所示.

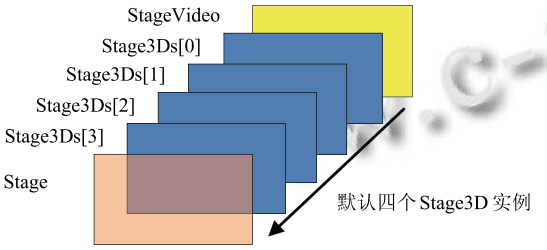

图 1　Stage3D 渲染序列

Stage3D 使用了可编程图形管线以实现对 GPU 的支持, 早年发布的 3D 图形加速管线为固定功能管线, 固定功能管线不够灵活, 它只将接收到的几何体数据作为输入, 通过管线单元处理这些数据, 然后将生成的最终渲染图像作为输出. 可编程图形管线引入了两个叫做顶点着色 (Vertex Shader)和片段着色(Fragment Shader)的新模块. 这两个模块是可以编程的. 用户可以通过写一段着色器代码, 用以影响渲染管线中发生的事情. 通过顶点着色器影响顶点如何变换以及通过片段着色器影响三角形像素如何渲染, 可以创建出固定功能管线不可能渲染出的惊艳效果.

Adobe 开发了图形汇编语言 AGAL 以操控可编程显卡. 编译器 AGALMiniAssembler 可以将字符串指令编译为 AGAL 二进制流, 再通过 context3D 上传给显卡的编程管线. 顶点以及片段的运算都是通过 AGAL 交由显卡来处理的.

在 Stage3D 中, 3D 场景是由一系列的 3D 对象组成的. 每个 3D 对象被定义为一系列的三角形. 而每一个三角形又被定义为一系列的顶点. 所有描述几何体的顶点被包装在一起组成一个叫做 Vertex Buffer 的结构. 这个结构包含所有与顶点相关的数据. 但 Vertex Buffer 还不足以定义一个几何体. 还需要一个额外的结构来从 Vertex Buffer 中提取顶点并把它们组装成三角形. 这个结构就叫做 Index Buffer. 几何体定义完成后需要用到着色器, 顶点着色器用来计算顶点从三维到二维的坐标投影. 而片段着色器用来计算顶点组成的三角形的填充颜色或者贴图, 用户可以自行编写顶点着色器和片段着色器并上传至显卡. 通过以上的流程便可创建出基本的 3D 场景.

3 基于3D引擎创建3D场景

直接使用 Stage3D 开发还是相当麻烦的. 一般情况下, 我们可以使用一些基于 Stage3D 的 3D 引擎. 常用的 3D 引擎有 Away3D、Alternativa3D、Flare3D 等. 它们都支持导入 3dsMAX、Collada 等模型; 拥有可视化编辑场景工具; 支持光照系统、动画、骨骼、摄像机系统等[4]. 本文主要研究了 Flare3D.

3.1 Flare3D 的基本结构和原理

使用 Flare3D 创建 3D 场景, Scene3D 是主元素, 包含并管理着所有的 3D 资源和对象. Scene3D 包含了一个 Clock3D 实例, 用来管理每帧画面的渲染刷新循环过程. Update 事件声明当前场景元素需要刷新, render 和 postRender 事件则提供了场景渲染过程前后的完全控制. 一旦场景建立, 使用 addChild 或 addChild FromFile 来向场景内添加 3d 对象, 该过程会发送对象加载进度 progress 事件以及对象加载完成 complete 事件, 以便控制从外部资源中加载 3D 模型的过程.

Scene3D 会创建一个 Camera3D 对象以便浏览场景, 画布的尺寸缺省情况下为当前舞台(Stage)的尺寸, 使用 setViewport 方法可以修改渲染区域的尺寸.

3.2 模型的制作

场景内的 3D 模型通常使用 3dsMax、SketchUp 等建模工具构建后导入场景, Flare3D 支持两种格式的 3D 模型, DAE 和 f3d.

DAE(Digital Asset Exchange)是 COLLADA 的模型文件. 基于 XML 交换标准, 有着十分优秀的移植性, 居于不同平台的实时引擎只要支持这个标准, 就可以实时表现三维模型. 目前我们常见的三维建模软件 3dsMax、Maya、SketchUp 都支持 DAE.

f3d 是 Flare3D 的私有格式, Flare3D 提供了多个版本的 3dsMax 导出插件, 可以将 3dsMax 模型导出为 f3d 文件. 和 DAE 相比, f3d 可以将资源(纹理信息等)嵌入到文件中, 并提供压缩、导出隐藏对象、导出动画等功能, 文件尺寸更小、加载速度更快, 部署更为方便.

3.3 动画的实现

模型的动画有两种方式

1. 动画回放. 在建模软件中可以为模型编辑动画, 3D 引擎提供了动画回放的功能.

首先我们为模型定制动画序列, 例如我们绘制了一个机柜, 并为它制作了开门和关门的动画, 其中 0-50 帧为开门. 50-100 帧为关门, 则我们可以这样设定:

```
model.addLabel( new Label3D( "open", 0, 50 ) );
model.addLabel( new Label3D( "close", 50, 100 ) );
```

设定完动画序列后就可以进行动画进行控制, 具体来说, 有以下几种方式:

play: 按顺序播放模型中的所有动画;

stop: 停止在当前动画帧上;

gotoAndPlay: 播放预先设定的动画序列, 可以设定为循环模式、乒乓模式或停止模式.

gotoAndStop: 将动画停止在某个指定帧上.

2. 实时控制的动画, 同样以机柜的开关为例, 我们可以获取到模型内部的每一个具体组成部分, 如柜门:

```
door = model.getChildByName( "door" );
```

然后我们就可以响应场景刷新事件, 控制门的旋转角度来生成动画.

```
door.setRotation( xRotation, yRotation, zRotation );
```

图 2 展示了一个机柜的不同状态.

图 2 动画的不同状态

3.4 碰撞检测的应用

Flare3D 提供了三种碰撞检测: 鼠标碰撞检测、光线碰撞检测和球形碰撞检测.

鼠标碰撞检测的应用: MouseCollision 类提供了鼠标的二维坐标与 3D 物体之间的碰撞检测手段, 使用此手段, 我们可以获得当前 3D 场景中位于鼠标位置处的所有平面以及他们的层次关系, 这些平面可以是一个物体的, 也可以是多个物体的, 借助此功能, 我们可以定制出各种复杂的鼠标交互方式.

光线碰撞检测的应用: RayCollision 类提供了 3D 虚拟光线和其他物体之间的碰撞检测手段, 虚拟光线能够测试与 3D 场景的对象是否相交. 如果测试成功, 将会提供关于碰撞的附加信息, 其中包括碰撞对象、碰撞的确切点、被碰撞平面的法线等信息. 例如可以提供 3D 模型的不均匀表面信息. 要实现人在不规则的路面上奔跑的功能就可以借助此手段.

球形碰撞检测的应用: SphereCollision 类是用于处理复杂几何图形碰撞的工具. 它本质上是一个可以与 3D 对象发生碰撞和相互作用的虚拟球体. 使用该球体可以检测它与周围物体之间的碰撞信息. 并生成以下三种方式的反应. 滑动模式提供了一种滑动响应, 观察点可以沿碰撞表面滑动; 固定模式产生一个静止响应, 观察点定位于准确的物体碰撞点; 相交模式则仅检查对象是否相交而不生成任何响应. 使用此手段我们可以控制漫游模式下观察点的位置, 以避免进入 3D 物体内部, 如果有需要, 还可以制造出观察点碰撞后沿物体表面滑动的效果.

3.5 交互功能

Flare3D 引擎提供了以下几种交互手段

Input3D 类来支持键盘和鼠标交互.

结合上面提到的碰撞检测我们可以定制交互功能, 例如我们需要实现在凹凸不平的平面上移动 3D 对象, 首先我们需要创建 RayCollision 对象, 并加入需要处理的 3D 平面 ruggedPlane.

```
var ray:RayCollision  = new RayCollision();
ray.addCollisionWith(ruggedPlane, false );
```

然后我们指定光线的起点 origin 和方向 dir, 并开始测试光线碰撞.

```
if (ray.test( origin, dir ) )
{
    var info:CollisionInfo = ray.data[0];
}
```

CollisionInfo 中包含了碰撞的位置和法向信息等, 依靠这些信息就可以实现物体在在凹凸不平的平面上的移动.

同时 3D 对象类本身也提供了拖拽方法, startDrag 和 stopDrag, 可以籍此实现常见的拖拽对象功能.

3.6 漫游

漫游分为两种方式: 方式 1 为观察点沿预先定制的路径移动, 从而形成动态的观察画面, 方式 2 为用户实时控制观察点移动, 类似于第一视角游戏. 下面我们分析一下两种漫游的具体实现方式.

方式 1:

首先由用户在 3D 场景上设定出一系列的漫游关键点, 所谓关键点, 也就是构成漫游路径的、变化方向较大的点, 也可以称之为特征点.

根据关键点计算出路径上每帧画面的具体观察点, 路径点的间距和渲染帧率为反比关系. 确定间距后在每两个相邻的关键点之间插值, 形成具体的观察点.

计算每个观察点上的观察角度. 需要注意的是, 观察的角度是随路径渐变的, 这样才能形成连续的观察画面, 因此我们在确定当前观察点的观察角度时就不能仅仅考虑下一个观察点位置, 而应该考虑到后续的一系列点的路径, 尤其是关键点. 否则就会出现这样的情况: 用户在某个关键点的观察角度发生突变, 导致观察画面发生跳动.

计算出上述信息后, 处理就较为简单了, 只需响应场景刷新事件, 按顺序为观察点设定坐标以及观察角度即可.

方式 2 的实现较为简单, 只需创建一个虚拟的观察者, 将摄像机添加为该观察者的附属对象, 使用交互方式控制观察者移动位置, 同时配合使用碰撞检测类 SphereCollision 即可实现. 图 3 展示了一个漫游状态下的场景.

图 3 漫游状态场景

4 Flash3D应用的设计和部署

和普通的 Flash 应用一样, 我们用 Flash Builder IDE 来开发 Flash3D 应用, 区别在于 Flash3D 应用必须编译为版本为 13 的 swf, 以及在嵌入 swf 的页面中将 WMODE 设置为"direct"以开启硬件加速功能. 作为 Web 应用, 它最终需要在 Web 服务器中部署, 根据实际业务的需要, 我们可以采用单一部署方式或数据交互方式. 如果该应用只是负责展示功能, 那我们可以采用单一部署方式, 只需将 Flash3D 应用(swf、html 等)以及所使用的资源(模型、图片、音频等)复制到部署环境中. 如果该应用需要与外部数据进行交互. 这时我们就需要采用数据交互方式. 借助于 BlazeDS 或 LiveCycleDataService 的 Java 远程调用和 Web 消息传递使得后台的 Java 应用程序和运行在浏览器上的 Flash 应用程序能够相互通信[5,6]. 下图展示了数据交互方式下 Flash3D 应用的设计架构.

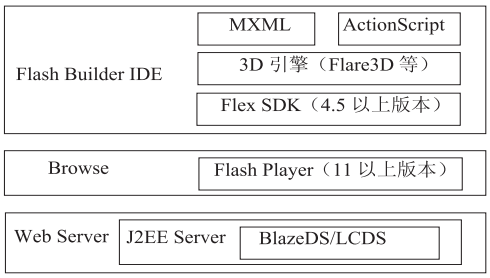

图 4 设计架构

5 结论

本文研究了基于 Flash 平台的、支持硬件加速的

(下转第 154 页)

(2) 自定义分片方式, 保证了片的最佳大小, 使集群中节点的计算量处于较均衡、正常的水平.

(3) 相比于[3,4]以及其他 Hadoop 应用, 省去了 Reduce 过程且 Map 进程相关无关, 算法流程更加简洁.

(4) 算法可适当地进行拓展以适应集群环境并提高性能.

Hadoop 作为刚刚兴起的计算平台, 还有很多特性需要去发掘和研究, 这些特性会有助于提升 Hadoop 平台上算法的表现, 以后会在这方面进一步研究.

参考文献

1 White T. Hadoop: The Definitive Guide. California. yahoopress. 2012: 1–15.

2 Islam S eds. An empirical distributed matrix multiplication algorithm to reduce time complexity. international association of engineers. Proc. of the International Multi Conference of Engineers and Computer Scientists 2009 Vol II.2009. HongKong. International Association of Engineers. 2009. 2171–2173.

3 曾大军.云平台下大型矩阵乘法运算处理方案设计.科技广场,2012,5:1–3.

4 张骏.一种基于 MapReduce 并行框架的大规模矩阵乘法运算的实现.计算机应用与软件,2012,6:1–4.

5 Dean J, Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters. USENIX. OSDI-2004. San Francisco, California, USA. USENIX. 2004. 137–150.

6 同济大学数学系.同济线性代数.北京:高等教育出版社,2007:47–48.

7 Xie J. Improving Performance of Hadoop Clusters. UMI Dissertation Express. 2011:6–7.

8 王谦.HADOOP 作业启动性能优化实践[硕士学位论文].北京:北京交通大学,2012:9–12.

(上接第 222 页)

3D 渲染 API Stage3D, 以及基于 Stage3D 的多个 3D 引擎, 并基于其中的一个引擎 Flare3D 实现了一个 B/S 架构的、包含编辑和展示功能的 3D 场景组件, 相对于早期的 Flash 平台, 该组件在 3D 场景的实时渲染性能方面得到了极大的提升,可以在普通显卡上支持百万级别的三角形面的实时渲染. 且使用简单, 部署方便, 可广泛应用于增强现实、立体场景、电子商务等高级应用中.

参考文献

1 何伟明,罗立宏.基于 Flash 的三维商品展示.广东工业大学学报(社会科学版),2009,B6:317–318.

2 王立英.基于 Flash 的在线三维商品展示系统的研究与实现[硕士学位论文].成都:电子科技大学,2011.

3 陈忻.FLASH 三维游戏开发探索[硕士学位论文].杭州:浙江大学,2008.

4 陈宁.一种基于 Away3d 的 Web 三维虚拟装配系统.黑龙江科技信息,2012,14:42–43.

5 杨逸文.基于 BlazeDS 的烟草移动服务综合监控系统.计算机应用与软件,2012,29(5):224–227.

6 吕海东,陆永林.基于 Flex 和 BlazeDS 推技术实现 WEB 方式实时监控系统.自动化技术与应用,2010,29(1):34–37.