Computer Science and Application 计算机科学与应用, 2019, 9(10), 1912-1919
Published Online October 2019 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2019.910214

Hans汉斯

Integration and Application of GL STUDIO in OSG

Guotao Zhu, Zhongyun Sun

Naval Aviation University, Qingdao Branch, Qingdao Shandong
Email: zealotsparc@163.com

Received: Oct. 1st, 2019; accepted: Oct. 16th, 2019; published: Oct. 23rd, 2019

Abstract

GL STUDIO is a powerful and virtual instrument development tool that can be used to create real-time, three-dimensional virtual instrumentation interactive simulation graphical interface. By analyzing the structure of GLS virtual instrument object, aimed for OSG rendering engine, GL STUDIO virtual instrument was transplanted to the OSG environment, the GL STUDIO virtual instrument was realized to load, display, update, and interact under OSG environment.

Keywords

Virtual Reality, Virtual Instrument, Flight Simulation

GL STUDIO虚拟仪表在OSG中的集成和应用

朱国涛，孙忠云

海军航空大学青岛校区，山东 青岛
Email: zealotsparc@163.com

收稿日期：2019年10月1日；录用日期：2019年10月16日；发布日期：2019年10月23日

摘 要

GL STUDIO是一套功能强大的虚拟仪表开发工具，可以用于创建实时的、三维的虚拟仪表交互仿真图形界面。本文通过分析GLS虚拟仪表的结构，针对OSG视景渲染引擎的特点，将GL STUDIO制作的虚拟仪表移植到了OSG视景色仿真引擎中，在OSG环境下实现了GL STUDIO虚拟仪表的加载、显示、更新和交互，为使用OSG视景仿真引擎构建飞行仿真训练系统虚拟座舱提供了一种新的方法。

文章引用: 朱国涛, 孙忠云. GL STUDIO 虚拟仪表在 OSG 中的集成和应用[J]. 计算机科学与应用, 2019, 9(10): 1912-1919.
DOI: 10.12677/csa.2019.910214

关键词

虚拟现实，虚拟仪表，飞行仿真

Copyright © 2019 by author(s) and Hans Publishers Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY).
http://creativecommons.org/licenses/by/4.0/
Open Access

1. 引言

Open Scene Graph (简称 OSG)是一个开源的高性能 3D 图形开发工具，它使用工业标准的 OpenGL 底层渲染 API，为 3D 应用程序提供了场景组织管理和图形渲染优化等功能，广泛应用于视景仿真、虚拟现实、科学可视化等领域[1] [2] [3]。OSG 可以运行在大多数操作系统之上，使用 OSG 进行开发能明显地减少费用，由于 OSG 优异的特性，越来越多的飞行仿真、舰船仿真等软件开始采用 OSG 进行开发[4] [5]，例如开源的飞行仿真软件 FlightGear、MAK 公司的三维态势软件 VR-Stealth 等，OSG 在世界可视化仿真市场的占有率逐年提高。

利用 OSG 完全能够满足一般仿真的需要，但是在飞行仿真领域，飞机座舱仪表的构建一般都需要使用专门的虚拟仪表软件完成，OSG 目前并没有提供专门的虚拟仪表开发工具，另一方面目前在国内飞行仿真领域广泛使用的虚拟仪表开发工具是 DISTI 公司的 GL STUIDO 软件，它提供了功能强大、开发便捷的虚拟仪表建模手段。如果能够将 OSG 的场景管理和图像渲染能力与 GL STUDIO 的虚拟仪表仿真建模能力结合起来，将有助于推动 OSG 在飞机仿真领域的应用。

2. 虚拟仪表工具 GL STUDIO 简介

GL STUDIO (简称 GLS)是 DISTI 公司开发的虚拟仪表开发工具，它是一个独立平台的快速原型工具，用来创建实时的、三维的虚拟仪表互动图形界面。使用 GL STUDIO 进行虚拟仪表仿真的过程如图 1 所示，用户可以使用 GLS 提供的可视化图形编辑器完成虚拟仪表面板、开关、按钮等元件的建模，然后使用 GLS 代码编辑器完成仪表内部逻辑的仿真，最后使用 GLS 代码生成器生成 C++/OpenGL 源代码。这些源代码既可以编译成独立的可执行程序(standalone 模式)，也可以编译成动态库(component 模式) [6] [7] [8]。

GL STUDIO 制作的虚拟仪表在其它视景引擎中使用时，通常编译成 component 模式，即将生成的仪表代码编译成一个动态链接库。因为 OSG 视景引擎底层与 GL STUDIO 一样使用 OpenGL 进行视景的渲染，因此从方法上来说，只要能够解决 GLS 虚拟仪表对象(即 GL STUDIO 制作包含虚拟仪表的动态链接库)的加载、显示、更新和交互问题，就能够将 GL STUDIO 虚拟仪表移植到 OSG 环境下。

在 Component 模式下，由 GLS 生成的虚拟仪表对象都是从 disti::ComponentBase 类派生的，它有几个重要的成员函数，其中 PreDraw 负责准备绘制环境，Draw 完成虚拟仪表绘制，GetExtents 返回虚拟仪表的包围盒，SetResource/GetResource 负责更新/获取对象的属性，HandleInput 负责处理用户鼠标或键盘交互事件，Calculate 函数(从 disti::Group 继承)负责对仪表状态进行计算和更新。此外在 GLS 虚拟仪表对象动态链接库中，还具有类似于 CreateRSOInterface1_xxxClass 形式的导出函数(其中 xxx 是虚拟仪表对象的名称)，调用这个函数将返回仪表对象的 RSO 远程共享指针，通过 RSO 指针的相关成员函数可以调用到仪表对象的 PreDraw、Draw、GetExtents、SetResource、HandleInput、Calculate 等成员函数。

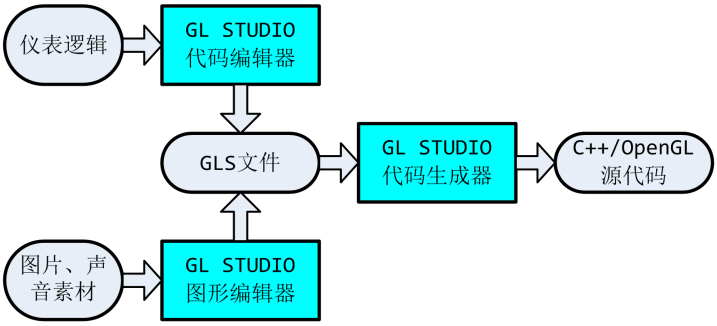

Figure 1. GL STUDIO virtual instrument development process
图 1. GL STUDIO 虚拟仪表开发过程

在其它 OpenGL 视景渲染环境中集成 GLS 虚拟仪表动态链接库，其调用顺序如图 2 所示，具体过程如下：

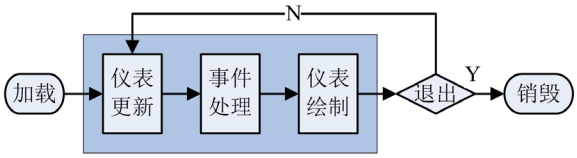

Figure 2. Call sequence of GL STUDIO virtual instrument dynamic library
图 2. GL STUDIO 虚拟仪表动态库调用顺序

1. 首先加载动态链接库，然后从中获取函数 CreateRSOInterface1_xxxClass 的地址，最后调用此函数获取仪表对象的 RSO 指针；

2. 在视景渲染的每个周期中，首先调用 Calculate 函数进行仪表状态计算和更新，然后调用 HandleInput 处理用户交互，最后调用 PreDraw 和 Draw 绘制虚拟仪表；

3. 程序退出时，销毁仪表对象并卸载动态链接库。

3. 基于 OSG 的 GLS 仪表集成

要完成 GLS 虚拟仪表在 OSG 场景中的更新、绘制和交互，还需要综合考虑到 OSG 场景结构和渲染流程，并解决以下问题：

1. 如何完成虚拟仪表的绘制？在 OSG 中，所有的可绘制对象都是从 osg::Drawable 类派生的，虚拟仪表的集成也需要从 osg::Drawable 类派生一个子类来完成。

2. 如何将 GLS 虚拟仪表对象组织到 OSG 场景图中？可以从 osg::Geode 几何节点类派生一个子类，osg::Geode 对象是 OSG 场景图中的叶子节点，负责几何体的绘制，一个 osg::Geode 对象可以包含一个或者多个 osg::Drawable 对象。

3. 如何更新虚拟仪表的属性和状态？虚拟仪表的更新可以通过设置对象的更新回调实现，可绘制对象的更新回调必须由 osg::Drawable::UpdateCallback 派生，在更新回调中调用 disti::ComponentBase::Calculate 函数即可计算和更新虚拟仪表的属性和状态。

4. 如何处理用户交互事件？交互事件处理可以自定义一个事件处理器完成，它由 osgGA::GUIEventHandler 派生，并重载其 handle 函数即可截获 OSG 事件并将其中的鼠标事件转换为 GLS 事件，然后调用

disti::ComponentBase::HandleInput 函数驱动虚拟仪表进行交互事件处理。

按照上述思路，设计插件的主要类图如图 3 所示。

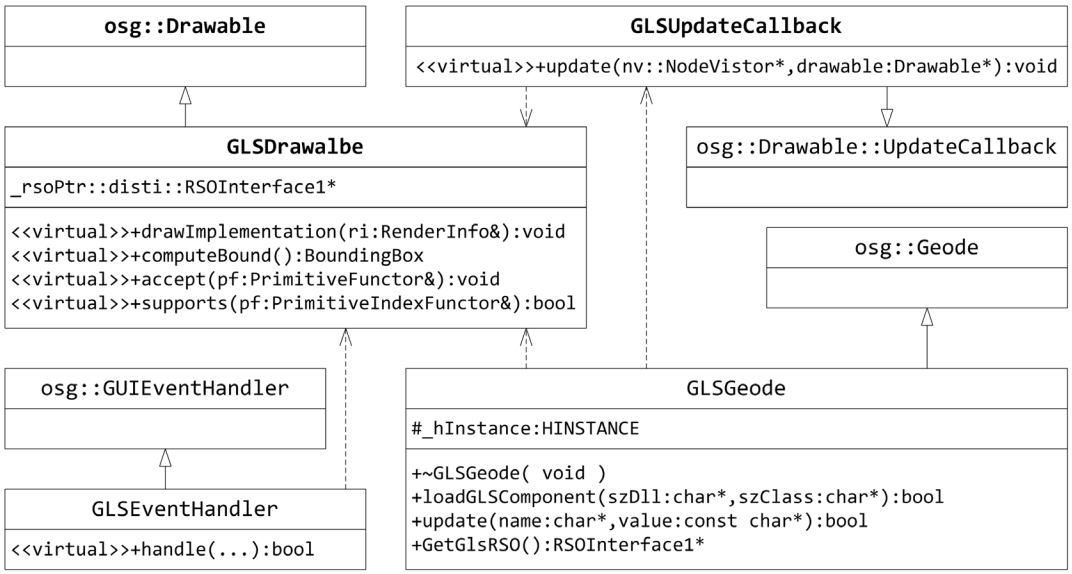

Figure 3. Class diagram of GL STUDIO virtual instrument integration plugin
图 3. GL STUDIO 虚拟仪表集成插件类图

OSG 视景引擎底层与 GL STUDIO 一样使用 OpenGL 进行视景的渲染，因此从方法上来说，只要能够解决 GLS 虚拟仪表对象(即 GL STUDIO 制作包含虚拟仪表的动态链接库)的加载、显示、更新和交互问题，就能够将 GL STUDIO 虚拟仪表移植到 OSG 环境下。上述类图中 GLSDrawable 负责虚拟仪表的绘制，GLSEventHandler 负责处理用户交互，即拦截 OSG 鼠标事件并将其转换为 GLS 鼠标事件并驱动动态库进行鼠标事件的处理，GLSUpdateCallback 负责更新虚拟仪表，GLSGeode 负责实现虚拟仪表动态库的加载和场景图的组织。

3.1. 虚拟仪表的加载

按照类图结构，虚拟仪表的加载需要从 osg::Geode 派生子类 GLSGeode 完成，GLSGeode 负责从动态库中加载GLS虚拟仪表并将虚拟仪表组织到场景图中，其加载过程在loadGLSComponent函数中实现，步骤如下：

1. 首先调用 Windows API 函数 LoadLibrary 加载指定的动态库，获取其句柄并保存到成员_hInstance 中；

2. 然后调用 GetProcAddress 函数获取 CreateRSOInterface1_xxxClass 导出函数的地址(其中 xxx 是虚拟仪表对象的名称)；

3. 最后调用上述导出函数获取指向仪表对象的 RSO 指针_pGlsRSO，然后使用此指针构造一个 GlsDrawable 对象并添加到 GLSGeode 对象中。

4. 在 GLSGeode 对象析构时，需要移出所有的 Drawable 对象并释放动态库。

3.2. 虚拟仪表的绘制

虚拟仪表的绘制在 GLSDrawable::drawImplementation 函数中完成，其绘制的关键代码如下：

```
void GLSDrawable::drawImplementation (osg::RenderInfo & renderInfo) const
{
  int viewport[4]={0,0,0,0};

  disti::RSOInterface1::MatrixD prj;
  disti::RSOInterface1::MatrixD mdl;

 // 获取当前的绘制参数
 ::glGetIntegerv( GL_VIEWPORT,viewport);
::glGetDoublev(GL_MODELVIEW_MATRIX,mdl._data);
::glGetDoublev(GL_PROJECTION_MATRIX, prj._data );
 // 构造 disti::RSOInterface1::PreDraw 参数
  disti::RSOInterface1::OpenGLMatricesmatrices(viewport,&prj,&mdl);
  disti::RSOInterface1::Culler culler;
  // 调用绘制
  _rsoPtr->PreDraw( matrices , culler);
  _rsoPtr->Draw();
  renderInfo.getState()->apply();
}
```

对于需要标注编号的公式，编号应写作“(1)”，不要写“Eq. (1)”或“Equation (1)”。

3.3. 虚拟仪表的更新

GLS 虚拟仪表对象的更新依赖于周期性的 Calculate 函数调用，这个调用在 OSG 环境下可以通过更新回调实现。首先从 osg::Drawable::UpdateCallBack 类派生 GlsUpdateCallBack，然后重载 update 函数，在 update 中通过虚拟仪表对象的 RSO 指针调用 Calculate 进行虚拟仪表的计算和更新。由于 OSG 在每一帧的渲染之前，都会调用更新回调对场景中的对象进行计算和更新，那么设置更新回调后 OSG 在渲染 GLS 虚拟仪表之前都会 GlsUpdateCallBack::update 更新虚拟仪表，其关键代码如下：

```
using namespace disti;
void GlsUpdateCallback::update(osg::NodeVisitor* nv,osg::Drawable* drawable)
{
static disti::Timer timer;

GLSDrawable* pGLSDrawable =dynamic_cast<GLSDrawable*>(drawable);
if( pGLSDrawable)
{
RSOInterface1* pGlsRSO =(RSOInterface1*)pGLSDrawable->GetGLSRSO();
pGlsRSO->Calculate(timer.ElapsedMilliseconds()/1000.0f);
}
}
```

更新过程为：如果传入的第二个参数即 osg::Drawable 对象指针是一个 GLSDrawable 对象的实例，则获取当前时间值(单位毫秒)并调用 Calculate 函数进行更新。

3.4. 交互事件的处理

GLS 虚拟仪表移植到 OSG 环境后，它并不能自动捕获并处理用户交互事件，因此如果需要它响应用户的交互事件，就需要在 OSG 中将交互事件转发给 GLS 虚拟仪表进行处理，这可以通过事件处理器实

现。

在 OSG 中，从 osgGA::GUIEventHandler 派生一个事件处理器 GlsEventHandler，重载其 handle 函数可以截获到所有的用户交互事件，但是截获到的事件是由 OSG 定义的，GLS 并不能理解这些事件的意义，因此必须将 OSG 事件转换成 GLS 本身的事件格式，然后才能由 GLS 处理。其关键实现代码如下：

```
using namespace disti::RSOInterface1;
using namespace disti::RSOInterface1::MouseEvent;
bool GlsEventCallback::handle(const osgGA::GUIEventAdapter& ea,osgGA::GUIActionAdapter& aa)
{
// 只处理鼠标按下\释放\拖动
switch( ea.getEventType() )
  {
  case osgGA::GUIEventAdapter::PUSH:
  case osgGA::GUIEventAdapter::RELEASE:
  case osgGA::GUIEventAdapter::DRAG:
    break;
  default: return false;
  }
  // 根据 OSG 事件构造 GLS 事件
  MouseEvent me;
  me._eventType = Event::EVENT_MOUSE;
  me._winLoc.Set( ea.getX( ), ea.getY() , 0.5 );

  switch( ea.getEventType() )
  {
  case osgGA::GUIEventAdapter::PUSH:
    me._eventSubtype= Event::MOUSE_DOWN;
    me._buttonMask= MouseButtonType(ea.getButton());
    break;
  case osgGA::GUIEventAdapter::RELEASE:
    me._eventSubtype= Event::MOUSE_UP;
    me._buttonMask= MouseButtonType(ea.getButton());
    break;
  case osgGA::GUIEventAdapter::DRAG:
    me._eventSubtype= Event::MOUSE_DRAG;
    me._buttonMask=MouseButtonType(ea.getButtonMask());
    break;
  default:
    return false;
  }

  // 由 GLS 虚拟仪表对事件进行处理
  ((disti::RSOInterface1*)pGlsRSO)->HandleInput(&me);
  return false;
}
```

4. 一些其它问题及解决方法

4.1. 鼠标事件的投影点击问题

鼠标事件的投影点击问题出现在 VEGA PRIME GL Studio Plug in version 2.6 中。此问题描述如图 4 所示，当用户在视口 a 位置点击鼠标时，用户的本意是按动仪表 1 左下角的 b 按钮，如果此时仪表 2 左下角的 c 按钮恰好处于仪表 1 后面并且在鼠标点击方向上，那么此时按钮 c 的动作也将被触发。在使用

GLS 虚拟仪表构建前后双座舱的飞机仿真系统时，这个问题特别严重，很可能导致在后舱进行鼠标操作时反而错误地触发前舱虚拟仪表的动作。

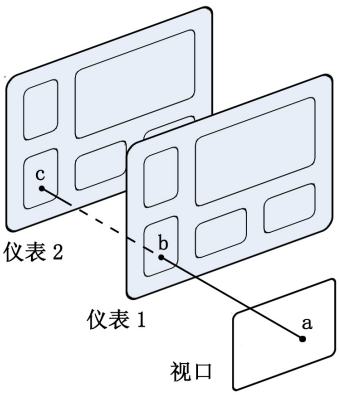

Figure 4. Click projection of mouse events
图 4. 鼠标事件的点击投影问题

这个问题解决方案如下：在分发鼠标事件之前，先以鼠标点击位置为端点，以垂直于视口向内的方向构造一条射线(即上图中以 a 为端点指向 c 点的射线)，然后使用此射线与场景中的对象(包含 GLS 虚拟仪表对象)进行相交检测并且按照交点位置的远近进行排序，如果最近的相交对象是 GLS 虚拟仪表对象，那么则将事件分发到这个最近的 GLS 虚拟仪表对象，否则事件将不会被分发和处理。

4.2. 场景中仪表的位置和姿态

将包含虚拟仪表的 GLSGeode 对象加入到场景图中，OSG 就可以渲染 GLS 虚拟仪表并与之交互。那么如何将虚拟仪表放置在场景中的合适位置呢？这可以通过合理的组织 OSG 场景图实现。某飞机仿真训练系统的场景组织如图 5 所示。其要点是首先构造一个 osg::Transform 变换节点，并将 GLSGeode 对象作为变换节点的子节点加入到场景图中，这样以来 OSG 在绘制 GLSGeode 对象之前，首先会将执行节点变换，节点变换将影响到 OpenGL 状态机当前的投影矩阵、模型视图矩阵和视口变换等，在 GLSGeode::drawImplementation 绘制过程中可以得到 OpenGL 投影矩阵、模型视图矩阵、视口变换的值，并以这些值为参数绘制虚拟仪表，这样虚拟仪表的位置、姿态就受到 osg::Transform 变换节点的影响。从而可以通过设置 osg::Transform 变换节点的位置、姿态、缩放等在场景中合理的放置虚拟仪表。

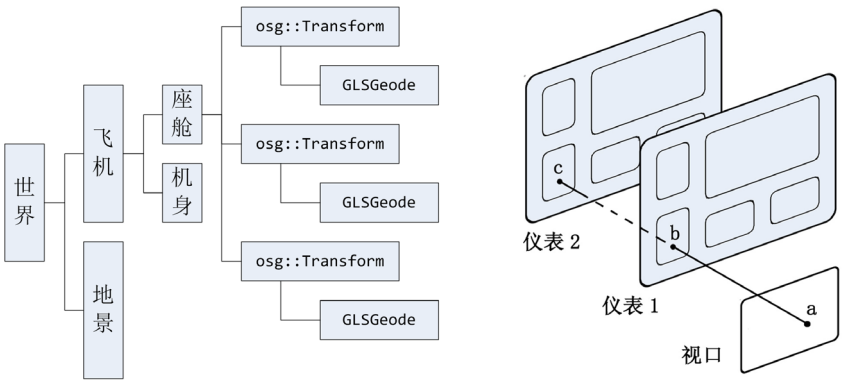

Figure 5. Example of scene diagram organization structure of GLS virtual instruments
图 5. 包含 GLS 虚拟仪表的场景图组织结构示例

5. 结论

Figure 6. Aircraft virtual cockpit constructed by GLS virtual instrument in OSG environment
图 6. 在 OSG 环境中使用 GLS 虚拟仪表构建的飞机虚拟座舱

伴随着开源视景仿真软件的兴起，越来越多的飞行仿真、舰船仿真等软件开始采用 OSG 进行开发，本文通过分析 GL STUDIO 虚拟仪表的结构，针对 Open Scene Graph 视景渲染引擎的特点，在 OSG 环境下实现了 GL STUDIO 虚拟仪表对象的加载、显示、更新和交互。在某型飞机飞行仿真训练系统中，我们首先使用 3D MAX 三维建模软件制作了包含座舱内部细节的飞机三维模型，然后使用 GL STUDIO 对飞机的主要仪表板进行了建模和仿真，这些仪表板编译成动态库后，在 OSG 中通过集成插件将虚拟仪表贴合到三维座舱模型中并进行交互驱动，其座舱仿真效果如图 6 所示。实践表明该方法能够从 GLS 开发工具生成的动态库中加载虚拟仪表并进行显示、更新和交互，其渲染效果、渲染效率能够满足飞机仿真、舰船仿真的要求。

参考文献

[1] Kessenich, J., Sellers, G. and Shreiner, D. (2016) OpenGL Programming Guide. 9th Edition, OpenGL Architecture Review Board, 7.

[2] Martz, P. (2007) Open Scene Graph Quick Start Guide. Skew Matrix Software LLC, Louisville, 13-31.

[3] 王锐, 钱学雷. Open Scene Graph 三维渲染引擎设计与实践[M]. 北京: 清华大学出版社, 2010.

[4] 孙艳丽, 王玲玲, 陈佳琪. 基于 GL Studio 的虚拟仪器仪表设计与仿真[J]. 系统仿真技术, 2015, 11(2): 151-155.

[5] 刘超慧, 韩晨, 魏家华. 基于 GL Studio 某型飞行训练模拟器导航界面的建模与仿真[J]. 舰船电子工程, 2018, 38(12): 114-117.

[6] Distributed Simulation Technology Inc., GL Studio Version 5.0 API Documentation. USA, 2014.

[7] Distributed Simulation Technology Inc., GL Studio User's Guide. Version 5.0 USA, 2014.

[8] 李建海, 何青洋, 孙艳丽. 基于 GL Studio 的航空虚拟仪表设计[J]. 计算机与数字工程, 2017, 45(5): 999-1002.