* **类的组成：**暂时来说，Noise3D库有如下几个类，所有跟3d场景有关系的class全部放在Noise3D命名空间里面，（2017.5.31）**现在每种资源或者场景节点都要有对应的管理对象，如GraphicObject有GraphicObjectManager**：

IRoot(2016.5)

ICamera

IScene

IAtmosphere(2015.9.18)

IMeshManager,IMesh(2016.7)

IRenderer

ILightManager,ILight(2016.7)

NoiseLineBuffer(2015.8.2)

IGraphicObjectManager(2016.7),IGraphicObject(2015.8.30)

IFileManager(2015.7.22)

Font套装(2015.11.9)

IMaterialManager(2015.8.20),IMaterial (2016.7) //-\_-。欧洲旅游完啦

下面这些就在Noise3D::Ut namespace 里面啦（自己弄的半吊子GUI就先不要了）.

IMeshSlicer(2015.7.25)

NoiseGUI套装(2015.10.23)

IInputEngine(2015.10.1)

ITimer

* **设计模式：(2016.8)**现在类的生存期管理要相对严格，所有对象必须由其父对象创建和销毁，这也就说明要涉及到对象的层次结构，这在下一节有图示。因为所有接口(interface)的构造和析构函数都是private的，所以普通用户是没有权限new或者定义一个Noise3D里的接口的权限（但是struct,即名字N\_开头的可以）。有一个专门设计的生存期管理模板类IFactory<>，这个类可以用Index, uid (unique identifier，唯一标识符，2016.8所用的uid依旧是std::string)来对资源进行create,GetPtr,Delete等操作，内置unorder\_map加速uid-access operation。因为用index来Access可能会不安全（因为原本的设计是Create了一个资源以后返回一个UINT index，可是这样的话如果有元素删除，那么这个index就不是想要的了，所以还是用一种唯一标识符来hash到对象会比较好，这样就不怕删除和创建操作了）。IRoot根对象的单例可以由一个全局函数GetRoot() (2016.8)来得到，然后再一层一层得得到想要的对象。例如IScene是IRoot的产品，即IScene只能由IRoot来创建而不能被new出来或者定义一个IScene类，所以**单例模式**就比较容易实现，因为对象的生存期管理的比较严格。这个时候，IScene的构造/析构函数就是私有的，而且IRoot 会public继承IFactory<IScene>，里面会有创建IScene指针的方法，而且IScene也要friend IFactory<IScene>，**这样IScene的构造函数就能只让IFactory access到**
* **场景资源层次结构：**

(备注：其中FontMgr内置GraphicObjMgr和TextureMgr, 也是由IScene给出单例