IFactory对象管理器

说是Factory不如说是一个Manager，用于管理子对象的生存周期，所以子对象也叫做Product。鉴于之前Noise3D里因为有不少对象之间是有联系的，例如Renderer必须得到各种资源（Mesh,Light,Texture,Camera等）才能进行渲染，所以就要进行一些”attach”。但是这种attach是先在用户自己创建对象，再手动Attach的，例如SetCamera,SetRenderer,SetTextureManager等，这样感觉也不太安全，用户也容易忘记这些事，所以给人感觉就是很不可控。所以后来开发了一个对象生存期管理模块IFactory。

所以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**