# Nexus Engine中的Device Lost处理

## 问题定义和分析

程序创建的D3D9 Device会在某些时候会和Adapter断开，导致所有的显存内的资源失效，这时候必须release所有D3DPOOL DEFAULT资源（D3DPOOL MANAGED的没关系），然后检测设备可以恢复时，调用device的Reset方法，然后重建之前释放掉的那些资源。可能导致Device Lost的事件包括：

1. 全屏模式下，窗口失去焦点；
2. 用户按Ctrl+Alt+Delete三键时；
3. 屏幕保护程序启动，以及系统进入待机状态；
4. 启动了其他全屏模式的D3D程序；

这个流程本身比较清楚，但是首先要在Device Lost时使用一个比较好的方式通知那些需要Release的资源对象；其次，真正的难点在于，恢复那些资源的内容。经过分析，可以把渲染用到的资源分为三类：

1. 渲染模块内部使用的，渲染过程中用到的资源：包括Render Targets，Shadow Buffer，Shader Effect等；
2. 引擎资源管理器管理的nresource派生类东西也管理一些渲染资源，例如nresource\_static\_mesh就管理一些渲染用的mesh；
3. Actor components管理的一些单个实例需要的渲染资源，包括动画用到的mesh等；

在上一代引擎F3D中使用了“统一处理”的方式，就是在底层的vertex buffer、index buffer等类中cache一份内容，感觉这样很浪费内存。最终决定，还是把device lost事件派发到上层的对象中，由上层对象去处理，包括释放资源，重建资源，以及恢复其内容。这样做的缺点就是把device lost这一渲染细节问题暴露到了上层，显得不是很优雅；优点就是每个对象可以针对自己的特点处理，最主要是“恢复内容”的操作！

## 具体设计思路

参考D3D的例子，以及D3DX接口的设计，将Device Lost事件的通知分为两步：

1. 在Device Lost时通知各对象释放资源；
2. 在Device reset之后，通知各对象重建资源，并恢复其内容；

经过前面分析，我们需要通知三类、多个对象，所以在nrender\_base中增加了注册/注销这两个事件的接口。函数回调使用了boost::function，以方便绑定到不同类的成员函数上。

在engine层中，主要是actor component和resource，这两种对象的处理起点函数分别绑定到了nengine类(最外层容器，level->actor->actor component)和nresource\_manager类中；

在nresource\_manager中因为要访问nresource\_cache中的所有对象，所以又使用了一个visitor模式去处理。

## 改进

一开始设计为上层的Actor Component，Resource，以及渲染底层的Render Resource都处理device lost，没有一个严格的区分哪些是必须上层做哪些是必须底层做，显得有点乱。特别是有的Render Resource由上层处理，有的需要自己处理。最后改为去掉Render Resource自身的device lost处理，全部由上层处理。