# Nexus Engine特效系统设计

## 前言

常见引擎中的特效系统大多局限于粒子系统，例如Unreal Engine 3就是实现了一套非常好的粒子系统，但是我们的思路与这些传统的特效系统大有不同。我们现在的设计思路的前身是由“刘业平”在《仙剑奇侠传3》项目中初创，后由房燕良在《功夫世界》、《龙online》项目中进行改造的一套以特效元素为基础的特效系统。现在又根据在《神兵传奇》项目中应用Unreal Engine 3的一些好的点进行整合而成的这样一套系统。

我们认为RPG游戏对于特效系统的要求较之FPS要高出很多。这最主要是因为RPG中特效往往要表达的是游戏虚拟世界中的一些概念，例如法术，瞬间传送，神界、鬼界气氛。在我们分析了大量的PC，PS，Xbox游戏的特效之后，发现传统的粒子系统只是我们RPG游戏特效的一小部分功能，我们还需要实现最常用的Billboard，球体，爆炸模拟等效果。当然，你也可以说这些是特殊的粒子发射器，但是我们的设计思路既然不局限于粒子，所以也就没有必要把这些东西往粒子上硬套了。

## 整体设计



图表 1

* 每种特效元素需要设计两个类，基类neffect\_element及其派生类用来管理这个特效元素的共享数据，这些数据由特效编辑器编辑获得；neffect\_element\_instance及其派生类管理特效元素实时更新时的状态，以及渲染需要用到的render mesh对象；
* 多个neffect\_element对象由一个nresource\_special\_effect对象管理；nresource\_special\_effect派生自nresource对象，由nresource\_manager按照资源方式统一管理；
* 多个neffect\_element\_instance对象由一个nspecial\_effect\_instance对象管理；nspecial\_effect\_instance派生自nprimitive\_component，按照actor component方式由actor管理；

### 可变数据抽象：Distribution



图表 2

特效最大的特点就是它自身管理的很多内部参数需要随时间变化，这些变化最终反映到渲染对象的形状、大小、颜色等可见元素上，来实现丰富的视觉效果。

根据上述需求，我们把特效元素（neffect\_element）管理的变量分类两类，一类为静态量，一般用来初始化；另外一类是动态量，这些量由ndistribution来封装，以实现“数值随时间变化”的抽象，目前有三种随时间变化的方式：

1. 常量（const）：不随时间而变；
2. 随机量（uniform）：在一个最大值和最小值之间取随机；
3. 曲线变化（curve）：由特效编辑编辑一个值随时间变化的曲线，在程序运行时，根据此曲线进行插值或者当前值（目前考虑使用线性插值即可）。

### 纹理坐标封装：Sub UV

 

图表 3

特效元素大量的需要程序生成几何体，这些几何体的位置、法线、纹理坐标都需要程序计算生成。有时我们需要一张完整的贴图，有时为了效率，我们往往需要把几张贴图拼到一张贴图中使用。为了实现这种灵活性，我们使用了nsub\_uv类来对uv坐标生成进行包装。

如上图所示，一个火焰粒子系统，可以通过设置sub uv来使得每个粒子只使用1/4的贴图。而在产生粒子的时候，粒子发射器可以控制逻辑在随机选择某个image\_index。

## 特效材质

todo

## 特效元素

### neffect\_quad

最简单的一个片，可以实现筒形billboard或者球形billboard功能，可是实现拖尾（每隔固定时间增加一个片）；

### neffect\_cylinder

筒形几何体；通过设置参数可以变形为带、星星几何体；