* **NoiseTextureManager：**

1. (2015.8.22)管理纹理。无效贴图的tex ID为-1。
2. (2015.9.24)内置Validate Texture ID的函数，可以检查ID是否越界，Shader resource view是否为对应的类型(common / cubemap/volumn)。
3. (2015.9.24)新增对CubeMap的支持。（暂时只能从设置好的dds中加载）
4. (2015.10.30)现在TextureObject会多一个vector用于在内存储存图片，我们叫这个是**内存副本**，在创建Texture的时候有一个BOOL可以选择是否创建内存副本。如果不用创建，直接D3DX11CreateSRV创建usage\_default的SRV的就好。但是要留一个内存副本的话，暂时的解决方案就是首先用D3DX11CreateTextureFromFIle创建一个Usage\_Staging的texture，然后Map()回来copy进vector里，之后用这个内存副本初始化一个usage\_default的texture并创建SRV。这里有个坑，用d3dx加载图片map回来之后RowPitch(Width)会有所变化，大概是有点取整的感觉，例如638px的宽会被拉伸成640,1238px被拉成1240px.
5. (2015.10.30)可以把有内存副本的Texture转换为灰度图和法线贴图。灰度图可以调rgb权重（还没归一化），法线图可以调高度场的起伏程度。
6. (2015.10.31)可以修改纹理的内存副本并更新到GPU。出于性能考虑建议先用UINT储存材质ID，不要在循环里用GetIndex （还有GetIndex以后用std::Map来实现Index-Name的配对）
7. (2015.11.10)由于字体纹理/纹理查询表被fontMgr委托由TexMgr管理，这些纹理不能被随便修改和删除，所以纹理应该有访问权限(ACCESS\_PERMISSION)，由FontMgr创建的纹理就应该由FontMgr来裁决而不能被用户操作影响，所以每个TextureObject都会有ACCESS\_PERMISSION属性，每次validateTextureID时如果PERMISSION不吻合就会返回INVALID\_TEXTURE\_ID。
8. (2015.11.18)GetTextureID的线性查找真是慢的吃屎，没什么事不要在循环里用GetTextureID。
9. (2016.2.22)统一使用std::string的名字来标识Texture，为了加快材质的查找和访问速度，特地添加了一个std::unordered<std::string,UINT>的哈希表，把texture名字映射到对应texture在vector里的index。一开始有想过只允许用string来访问texture（即string-N\_TextureObject的pair），但是如果要批量操作好像还是有点烦，因为并没有实现迭代器，所以还是允许用index对texture进行访问。但是material里就变成用string来标识各种map，因为记录material创建时候各种map的Index并不靠谱，index是会随着资源的增加和删除而改变，所以index是不靠谱的，要用唯一标识符(uid)，这里我选择使用名字作为唯一标识符。Noise Mat Mgr也使用类似的，有个哈希表，用名字作为uid管理资源。