### 7.1 Cg 关键字

Cg 中的关键字很多都是照搬 C\C++中的关键字，Cg中也创造了一系列独特的关键字，这些关键字不但用于指定输入图元的数据含义（是位置信息，还是法向量信息），本质也则对应着这些图元数据存放的硬件资源（寄存器或者纹理），称之为语义词（Semantics），通常也根据其用法称之为绑定语义词（binding semantics）。

### 7.2 uniform

Varying inputs,即数据流输入图元信息的各种组成要素。从应用程序输入到 GPU 的数据除了顶点位置数据，还有顶点的法向量数据，纹理坐标数据等。Cg语言提供了一组语义词，用以表明参数是由顶点的哪些数据初始化的。

Uniform inputs，表示一些与三维渲染有关的离散信息数据，这些数据通常由应用程序传入，并通常不会随着图元信息的变化而变化，如材质对光的反射信息、运动矩阵等。Uniform 修辞一个参数，表示该参数的值由**外部**应用程序初始化并传入。

“外部”的含义通常是用 OpenGL 或者 DirectX 所编写的应用程序。

uniform 修辞的变量的值是从外部传入的，所以在 Cg 程序（顶点程序和片段程序）中通常使用 uniform 参数修辞函数形参，不容许声明一个用 uniform 修辞的局部变量！（Error C5056:‘uniform’ not allowed on local variable）

### 7.3 const

Cg 语言也提供 const 修辞符，与 C\C++中含义一样，被 const 所修辞的变量在初始化之后不能再去改变它的值。

const 修辞符与 uniform 修辞符是相互独立的，对一个变量既可以单独使用const 或者 uniform，也可以同时使用。

### 7.4 输入\输出修辞符（in\out\inout)

参数传递是指：函数调用实参值初始化函数形参的过程。

参数传递：1、“值传递（pass-by-value）” 2、“引用传递（pass-by-reference）”

按值传递时，函数不会访问当前调用的实参，函数体处理的是实参的拷贝，也就是形参，所以形参值的改变不会影响实参值；

引用传递时，函数接收的是实参的存放地址，函数体中改变的是实参的值。（C\C++采取指针机制构建引用传递，所以通常引用传递也称为“指针传递”。）

指针机制并不被 GPU 硬件所支持，所以Cg 语言采用不同的语法修辞符来区别“值传递”和“引用传递”。这些修辞符分别为：

1. in: 修辞一个形参只是用于输入，进入函数体时被初始化，且该形参值的改变不会影响实参值，这是典型的**值传递**方式。

2. out: 修辞一个形参只是用于输出的，进入函数体时并没有被初始化，这种类型的**形参**一般是一个函数的运行结果； （也可以使用 return 语句来代替 out 修辞符的使用。）
3. inout: 修辞一个形参既用于输入也用于输出，这是典型的**引用传递。**

void myFunction(out float x); //形参 x，只是用于输出
void myFunction(inout float x); //形参 x，即用于输入时初始化，也用于输出数据
void myFunction(in float x); //形参 x，只是用于输入
**void myFunction(float x); /等价与 in float x，这种用法和 C\C++完全一致**