**Appendix A - CG Function List**

**abs( x )**  
Absolute value of x .

**acos( x )**  
Arccos of x in range [0, pi], x in [-1, 1].

**all( x )**  
true if **all** components of x != 0 - false otherwise.

**any( x )**  
true if **any** component of x != 0 - false otherwise.

**asin( x )**  
Arcsin of x in range [-pi/2,pi/2]; x in [-1, 1].

**atan( x )**  
Arctan of x in range [-pi/2, pi/2].

**atan2( y , x )**  
Arctan y/x in range [-pi,pi].

**ceil( x )**  
Smallest integer not less than x .

**clamp( x , a , b )**  
x clamped to the range [a,b]: a if x<a, b if x>b else returns x.

**cos( x )**  
Cos of x.

**cosh( x )**  
Hyperbolic cos of x .

**cross( A , B )**  
Cross product of vectors A and B (A and B must be three-component).

**ddx( a )**  
Partial derivative of a with respect to screen-space x coordinate

**ddy( a )**  
Partial derivative of a with respect to screen-space y coordinate

**debug(float4 x)**  
If compiler DEBUG option is enabled, causes shader to halt with x copied to the COLOR output - otherwise it does nothing.

**degrees( x )**  
convert radians to degrees.

**determinant( x )**  
Determinant of matrix x.

**distance( p1 , p2 )**  
Pythagorean distance between p1 and p2.

**dot( A, B )**  
Dot product of A and B.

**exp( x )**  
Exponential func: ex.

**exp2( x )**  
Exponential function 2x.

**faceforward( N, I, Ng )**  
N if dot( Ng , I ) < 0 otherwise (-N).

**floor( x )**  
Largest integer not greater than x.

**fmod( x , y )**  
Remainder of x/y (with the same sign as x). If y==0, result is undefined.

**frac( x )**  
Fractional part of x.

**frexp( x , out exp )**  
Splits x into normalized fraction in the interval [, 1), which is returned, and a power of 2, which is stored in exp . If x==0, both parts of the result are 0.

**isfinite( x )**  
Returns true if x is finite.

**isinf( x )**  
Returns true if x is infinite.

**isnan( x )**  
Returns true if x is NaN.

**ldexp( x , n )**  
x \* 2n.

**length( v )**  
Pythagorean length of vector.

**lerp( a , b , f )**  
Linear interpolation: (1 - f)\* a + b \* f. f can be a vector of same length as a & b.

**lit( NdotL , NdotH , m )**  
Computes lighting coefficients (m is the shininess).  
Returns: float4: .x=ambient (always 1), .y=diffuse (0 if N.L < 0), .z=specular (0 if N.L<0 or N.H<0); .w==1.0.

**log( x )**  
Natural logarithm (x>0).

**log2( x )**  
Log Base 2 (x>0).

**log10( x )**  
Log Base 10 (x>0).

**max( a , b )**  
Maximum of a and b .

**min( a , b )**  
Minimum of a and b .

**modf( x , out ip )**  
Splits x into int and frac parts (each with the same sign as x). Stores the int part in ip and returns the fractional part.

**mul( M , N )**  
Matrix product of matrix M and matrix N: If M has size AxB, and N has size BxC, returns an AxC.

**mul( M , v )**  
Product of matrix M and column vector v: If M has size AxB, and v is Bx1, returns an Ax1.

**mul( v , M )**  
Product of row vector v and matrix M: If v is a 1xA and M is AxB, returns a 1xB.

**noise( x )**  
Noise function. The returned value is between 0 and 1, and is always the same for a given input value.

**normalize( v )**  
Normalize v.

**pow( x , y )**  
xy

**radians( x )**  
Converts degrees to radians.

**reflect( I , N )**  
Computes reflection of I in a plane with surface normal N. (three-component vectors only).

**refract( I , N , eta )**  
Computes refraction of I in a plane with surface normal N and refractive index eta. Returns (0,0,0) for total internal reflection. (three-component vectors only).

**round( x )**  
Closest integer to x.

**rsqrt( x )**  
Reciprocal square root of x (x>0).

**saturate( x )**  
Clamps x to [0, 1].

**sign( x )**  
Sign of x (returns -1, 0 or +1)

**sin( x )**  
Sin of x.

**sincos(float x , out s , out c )**  
s is set to sin(x), and c to cos(x).

**sinh( x )**  
Hyperbolic sine of x.

**smoothstep( min , max , x )**  
For values of x between min and max , returns a smoothly varying value that ranges from 0 at x==min to 1 at x==max. x is clamped to the range [min,max] and then returns 2\*((x-min)/(max-min))3+3\*((x-min)/(max-min))2

**step( a , x )**  
0 if x<a, 1 if x>=a

**sqrt( x )**  
Square root of x (x>0)

**tan( x )**  
Tan of x.

**tanh( x )**  
Hyperbolic tan of x.

**tex1D(sampler1D tex , float s )**  
1D nonprojective texture lookup

**tex1D(sampler1D tex , float s , float dsdx , float dsdy )**  
1D nonprojective texture lookup with derivatives

**tex1D(sampler1D tex , float2 sz )**  
1D nonprojective depth compare texture lookup

**tex1D(sampler1D tex , float2 sz , float dsdx , float dsdy )**  
1D nonprojective depth compare texture lookup with derivatives

**tex1Dproj(sampler1D tex , float2 sq )**  
1D projective texture lookup

**tex1Dproj(sampler1D tex , float3 szq )**  
1D projective depth compare texture lookup

**tex2D(sampler2D tex , float2 s )**  
2D nonprojective texture lookup

**tex2D(sampler2D tex , float2 s , float2 dsdx , float2 dsdy )**  
2D nonprojective texture lookup with derivatives

**tex2D(sampler2D tex , float3 sz )**  
2D nonprojective depth compare texture lookup

**tex2D(sampler2D tex , float3 sz , float2 dsdx , float2 dsdy )**  
2D nonprojective depth compare texture lookup with derivatives

**tex2Dproj(sampler2D tex , float3 sq )**  
2D projective texture lookup

**tex2Dproj(sampler2D tex , float4 szq )**  
2D projective depth compare texture lookup

**texRECT(samplerRECT tex , float2 s )**  
2D nonprojective texture rectangle texture lookup (OpenGL)

**texRECT(samplerRECT tex , float2 s , float2 dsdx , float2 dsdy )**  
2D nonprojective texture rectangle texture lookup with derivatives (OpenGL)

**texRECT(samplerRECT tex , float3 sz )**  
2D nonprojective texture rectangle depth compare texture lookup (OpenGL)

**texRECT(samplerRECT tex , float3 sz , float2 dsdx , float2 dsdy )**  
2D nonprojective depth compare texture lookup with derivatives (OpenGL)

**texRECTproj(samplerRECT tex , float3 sq )**  
2D texture rectangle projective texture lookup (OpenGL)

**texRECTproj(samplerRECT tex , float3 szq )**  
2D texture rectangle projective depth compare texture lookup (OpenGL)

**tex3D(sampler3D tex , float3 s )**  
3D nonprojective texture lookup

**tex3D(sampler3D tex , float3 s , float3 dsdx , float3 dsdy )**  
3D nonprojective texture lookup with derivatives

**tex3Dproj(sampler3D tex , float4 sq )**  
3D projective texture lookup

**texCUBE(samplerCUBE tex , float3 s )**  
Cubemap nonprojective texture lookup

**texCUBE(samplerCUBE tex , float3 s , float3 dsdx , float3 dsdy )**  
Cubemap nonprojective texture lookup with derivatives

**texCUBEproj(samplerCUBE tex , float4 sq )**  
Cube map projective texture lookup (ignores q)

**transpose( M )**  
transpose of matrix M

**NOTES:**

* s indicates a one-, two-, or three-component texture coordinate.
* z indicates a depth comparison value for shadow map lookups.
* q indicates a perspective value, and is used to divide the texture coordinate ( s ) before the texture lookup is performed.

When you use the texture functions that allow specifying a depth comparison value, the associated texture unit must be configured for depth-compare texturing. Otherwise, no depth comparison will actually be performed.

Reference: https://www.sjbaker.org/wiki/index.php?title=Concise\_Cg\_built-in\_function\_table