# Built-In Functions (continued)

## Common Functions (continued)

| | |
|---|---|
| T **smoothstep**(T *edge0*, T *edge1*, T *x*);<br>T **smoothstep**(float *edge0*, float *edge1*, T *x*); | clamp and smooth |
| TB **isnan**(T *x*); | true if *x* is a NaN |
| TB **isinf**(T *x*); | true if *x* is positive or negative infinity |
| TI **floatBitsToInt**(T *value*);<br>TU **floatBitsToUint**(T *value*); | highp integer, preserving float bit level representation |
| T **intBitsToFloat**(TI *value*);<br>T **uintBitsToFloat**(TU *value*); | highp float, preserving integer bit level representation |

## Floating-point Pack and Unpack Functions [8.4]

| | |
|---|---|
| uint **packSnorm2x16**(vec2 *v*);<br>uint **packUnorm2x16**(vec2 *v*); | convert two floats to fixed point and pack into an integer |
| vec2 **unpackSnorm2x16**(uint *p*);<br>vec2 **unpackUnorm2x16**(uint *p*); | unpack fixed point value pair into floats |
| uint **packHalf2x16**(vec2 *v*); | convert two floats into half-precision floats and pack into an integer |
| vec2 **unpackHalf2x16**(uint *v*); | unpack half value pair into full floats |

## Geometric Functions [8.5]

These functions operate on vectors as vectors, not component-wise. T is float, vec2, vec3, vec4.

| | |
|---|---|
| float **length**(T *x*); | length of vector |
| float **distance**(T *p0*, T *p1*); | distance between points |
| float **dot**(T *x*, T *y*); | dot product |
| vec3 **cross**(vec3 *x*, vec3 *y*); | cross product |
| T **normalize**(T *x*); | normalize vector to length 1 |
| T **faceforward**(T *N*, T *I*, T *Nref*); | returns *N* if **dot**(*Nref*, *I*) < 0, else *-N* |
| T **reflect**(T *I*, T *N*); | reflection direction *I* - 2 * **dot**(*N*,*I*) * *N* |
| T **refract**(T *I*, T *N*, float *eta*); | refraction vector |

## Matrix Functions [8.6]

Type mat is any matrix type.

| | |
|---|---|
| mat **matrixCompMult**(mat *x*, mat *y*); | multiply *x* by *y* component-wise |
| mat2 **outerProduct**(vec2 *c*, vec2 *r*);<br>mat3 **outerProduct**(vec3 *c*, vec3 *r*);<br>mat4 **outerProduct**(vec4 *c*, vec4 *r*); | linear algebraic column vector * row vector |
| mat2x3 **outerProduct**(vec3 *c*, vec2 *r*);<br>mat3x2 **outerProduct**(vec2 *c*, vec3 *r*);<br>mat2x4 **outerProduct**(vec4 *c*, vec2 *r*);<br>mat4x2 **outerProduct**(vec2 *c*, vec4 *r*);<br>mat3x4 **outerProduct**(vec4 *c*, vec3 *r*);<br>mat4x3 **outerProduct**(vec3 *c*, vec4 *r*); | linear algebraic column vector * row vector |
| mat2 **transpose**(mat2 *m*);<br>mat3 **transpose**(mat3 *m*);<br>mat4 **transpose**(mat4 *m*);<br>mat2x3 **transpose**(mat3x2 *m*);<br>mat3x2 **transpose**(mat2x3 *m*);<br>mat2x4 **transpose**(mat4x2 *m*);<br>mat4x2 **transpose**(mat2x4 *m*);<br>mat3x4 **transpose**(mat4x3 *m*);<br>mat4x3 **transpose**(mat3x4 *m*); | transpose of matrix *m* |
| float **determinant**(mat2 *m*);<br>float **determinant**(mat3 *m*);<br>float **determinant**(mat4 *m*); | determinant of matrix *m* |
| mat2 **inverse**(mat2 *m*);<br>mat3 **inverse**(mat3 *m*);<br>mat4 **inverse**(mat4 *m*); | inverse of matrix *m* |

## Vector Relational Functions [8.7]

Compare *x* and *y* component-wise. Input and return vector sizes for a particular call must match. Type bvec is bvec*n*; vec is vec*n*; ivec is ivec*n*; uvec is uvec*n*; (where *n* is 2, 3, or 4). T is union of vec and ivec.

| | |
|---|---|
| bvec **lessThan**(T *x*, T *y*);<br>bvec **lessThan**(uvec *x*, uvec *y*); | x < y |
| bvec **lessThanEqual**(T *x*, T *y*);<br>bvec **lessThanEqual**(uvec *x*, uvec *y*); | x <= y |
| bvec **greaterThan**(T *x*, T *y*);<br>bvec **greaterThan**(uvec *x*, uvec *y*); | x > y |
| bvec **greaterThanEqual**(T *x*, T *y*);<br>bvec **greaterThanEqual**(uvec *x*, uvec *y*); | x >= y |
| bvec **equal**(T *x*, T *y*);<br>bvec **equal**(bvec *x*, bvec *y*);<br>bvec **equal**(uvec *x*, uvec *y*); | x == y |
| bvec **notEqual**(T *x*, T *y*);<br>bvec **notEqual**(bvec *x*, bvec *y*);<br>bvec **notEqual**(uvec *x*, uvec *y*); | x!= y |
| bool **any**(bvec *x*); | true if any component of *x* is true |
| bool **all**(bvec *x*); | true if all components of *x* are true |
| bvec **not**(bvec *x*); | logical complement of *x* |

## Texture Lookup Functions [8.8]

The function textureSize returns the dimensions of level *lod* for the texture bound to sampler, as described in [2.11.9] of the OpenGL ES 3.0 specification, under "Texture Size Query". The initial "g" in a type name is a placeholder for nothing, "i", or "u".

| | |
|---|---|
| highp ivec{2,3} | **textureSize**(gsampler{2,3}D *sampler*, int *lod*); |
| highp ivec2 | **textureSize**(gsamplerCube *sampler*, int *lod*); |
| highp ivec2 | **textureSize**(sampler2DShadow *sampler*, int *lod*); |
| highp ivec2 | **textureSize**(samplerCubeShadow *sampler*, int *lod*); |
| highp ivec3 | **textureSize**(sampler2DArray *sampler*, int *lod*); |
| highp ivec3 | **textureSize**(sampler2DArrayShadow *sampler*, int *lod*); |

Texture lookup functions using samplers are available to vertex and fragment shaders. The initial "g" in a type name is a placeholder for nothing, "i", or "u".

| | |
|---|---|
| gvec4 | **texture**(gsampler{2,3}D *sampler*, vec{2,3} *P* [, float *bias*]); |
| gvec4 | **texture**(gsamplerCube *sampler*, vec3 *P* [, float *bias*]); |
| float | **texture**(sampler2DShadow *sampler*, vec3 *P* [, float *bias*]); |
| float | **texture**(samplerCubeShadow *sampler*, vec4 *P* [, float *bias*]); |
| gvec4 | **texture**(gsampler2DArray *sampler*, vec3 *P* [, float *bias*]); |
| float | **texture**(sampler2DArrayShadow *sampler*, vec4 *P*); |
| gvec4 | **textureProj**(gsampler2D *sampler*, vec{3,4} *P* [, float *bias*]); |
| gvec4 | **textureProj**(gsampler3D *sampler*, vec4 *P* [, float *bias*]); |
| float | **textureProj**(sampler2DShadow *sampler*, vec4 *P* [, float *bias*]); |
| gvec4 | **textureLod**(gsampler{2,3}D *sampler*, vec{2,3} *P*, float *lod*); |
| gvec4 | **textureLod**(gsamplerCube *sampler*, vec3 *P*, float *lod*); |
| float | **textureLod**(sampler2DShadow *sampler*, vec3 *P*, float *lod*); |
| gvec4 | **textureLod**(gsampler2DArray *sampler*, vec3 *P*, float *lod*); |
| gvec4 | **textureOffset**(gsampler2D *sampler*, vec2 *P*, ivec2 *offset* [, float *bias*]); |
| gvec4 | **textureOffset**(gsampler3D *sampler*, vec3 *P*, ivec3 *offset* [, float *bias*]); |
| float | **textureOffset**(sampler2DShadow *sampler*, vec3 *P*, ivec2 *offset* [, float *bias*]); |
| gvec4 | **textureOffset**(gsampler2DArray *sampler*, vec3 *P*, ivec2 *offset* [, float *bias*]); |
| gvec4 | **texelFetch**(gsampler2D *sampler*, ivec2 *P*, int *lod*); |
| gvec4 | **texelFetch**(gsampler3D *sampler*, ivec3 *P*, int *lod*); |
| gvec4 | **texelFetch**(gsampler2DArray *sampler*, ivec3 *P*, int *lod*); |
| gvec4 | **texelFetchOffset**(gsampler2D *sampler*, ivec2 *P*, int *lod*, ivec2 *offset*); |
| gvec4 | **texelFetchOffset**(gsampler3D *sampler*, ivec3 *P*, int *lod*, ivec3 *offset*); |
| gvec4 | **texelFetchOffset**(gsampler2DArray *sampler*, ivec3 *P*, int *lod*, ivec2 *offset*); |
| gvec4 | **textureProjOffset**(gsampler2D *sampler*, vec3 *P*, ivec2 *offset* [, float *bias*]); |
| gvec4 | **textureProjOffset**(gsampler2D *sampler*, vec4 *P*, ivec2 *offset* [, float *bias*]); |
| gvec4 | **textureProjOffset**(gsampler3D *sampler*, vec4 *P*, ivec3 *offset* [, float *bias*]); |
| float | **textureProjOffset**(sampler2DShadow *sampler*, vec4 *P*, ivec2 *offset* [, float *bias*]); |

## Texture Lookup Functions (continued)

| | |
|---|---|
| gvec4 | **textureLodOffset**(gsampler2D *sampler*, vec2 *P*, float *lod*, ivec2 *offset*); |
| gvec4 | **textureLodOffset**(gsampler3D *sampler*, vec3 *P*, float *lod*, ivec3 *offset*); |
| float | **textureLodOffset**(sampler2DShadow *sampler*, vec3 *P*, float *lod*, ivec2 *offset*); |
| gvec4 | **textureLodOffset**(gsampler2DArray *sampler*, vec3 *P*, float *lod*, ivec2 *offset*); |
| gvec4 | **textureProjLod**(gsampler2D *sampler*, vec3 *P*, float *lod*); |
| gvec4 | **textureProjLod**(gsampler2D *sampler*, vec4 *P*, float *lod*); |
| gvec4 | **textureProjLod**(gsampler3D *sampler*, vec4 *P*, float *lod*); |
| float | **textureProjLod**(sampler2DShadow *sampler*, vec4 *P*, float *lod*); |
| gvec4 | **textureProjLodOffset**(gsampler2D *sampler*, vec3 *P*, float *lod*, ivec2 *offset*); |
| gvec4 | **textureProjLodOffset**(gsampler2D *sampler*, vec4 *P*, float *lod*, ivec2 *offset*); |
| gvec4 | **textureProjLodOffset**(gsampler3D *sampler*, vec4 *P*, float *lod*, ivec3 *offset*); |
| float | **textureProjLodOffset**(sampler2DShadow *sampler*, vec4 *P*, float *lod*, ivec2 *offset*); |
| gvec4 | **textureGrad**(gsampler2D *sampler*, vec2 *P*, vec2 *dPdx*, vec2 *dPdy*); |
| gvec4 | **textureGrad**(gsampler3D *sampler*, vec3 *P*, vec3 *dPdx*, vec3 *dPdy*); |
| gvec4 | **textureGrad**(gsamplerCube *sampler*, vec3 *P*, vec3 *dPdx*, vec3 *dPdy*); |
| float | **textureGrad**(sampler2DShadow *sampler*, vec3 *P*, vec2 *dPdx*, vec2 *dPdy*); |
| float | **textureGrad**(samplerCubeShadow *sampler*, vec4 *P*, vec3 *dPdx*, vec3 *dPdy*); |
| gvec4 | **textureGrad**(gsampler2DArray *sampler*, vec3 *P*, vec2 *dPdx*, vec2 *dPdy*); |
| float | **textureGrad**(sampler2DArrayShadow *sampler*, vec4 *P*, vec2 *dPdx*, vec2 *dPdy*); |
| gvec4 | **textureGradOffset**(gsampler2D *sampler*, vec2 *P*, vec2 *dPdx*, vec2 *dPdy*, ivec2 *offset*); |
| gvec4 | **textureGradOffset**(gsampler3D *sampler*, vec3 *P*, vec3 *dPdx*, vec3 *dPdy*, ivec3 *offset*); |
| float | **textureGradOffset**(sampler2DShadow *sampler*, vec3 *P*, vec2 *dPdx*, vec2 *dPdy*, ivec2 *offset*); |
| gvec4 | **textureGradOffset**(gsampler2DArray *sampler*, vec3 *P*, vec2 *dPdx*, vec2 *dPdy*, ivec2 *offset*); |
| float | **textureGradOffset**(sampler2DArrayShadow *sampler*, vec4 *P*, vec2 *dPdx*, vec2 *dPdy*, ivec2 *offset*); |
| gvec4 | **textureProjGrad**(gsampler2D *sampler*, vec3 *P*, vec2 *dPdx*, vec2 *dPdy*); |
| gvec4 | **textureProjGrad**(gsampler2D *sampler*, vec4 *P*, vec2 *dPdx*, vec2 *dPdy*); |
| gvec4 | **textureProjGrad**(gsampler3D *sampler*, vec4 *P*, vec3 *dPdx*, vec3 *dPdy*); |
| float | **textureProjGrad**(sampler2DShadow *sampler*, vec4 *P*, vec2 *dPdx*, vec2 *dPdy*); |
| gvec4 | **textureProjGradOffset**(gsampler2D *sampler*, vec3 *P*, vec2 *dPdx*, vec2 *dPdy*, ivec2 *offset*); |
| gvec4 | **textureProjGradOffset**(gsampler2D *sampler*, vec4 *P*, vec2 *dPdx*, vec2 *dPdy*, ivec2 *offset*); |
| gvec4 | **textureProjGradOffset**(gsampler3D *sampler*, vec4 *P*, vec3 *dPdx*, vec3 *dPdy*, ivec3 *offset*); |
| float | **textureProjGradOffset**(sampler2DShadow *sampler*, vec4 *P*, vec2 *dPdx*, vec2 *dPdy*, ivec2 *offset*); |

## Fragment Processing Functions [8.9]

Approximated using local differencing.

| | |
|---|---|
| T **dFdx**(T *p*); | Derivative in *x* |
| T **dFdy**(T *p*); | Derivative in *y* |
| T **fwidth**(T *p*); | **abs** (**dFdx** (*p*)) + **abs** (**dFdy** (*p*)); |