OpenGL ESシェーディング言語

OpenGL ES Shading Language

C.1 GLSL ES言語リファレンス

この付録は、WebGL用のシェーダをプログラムするために使われるGLSL ES言語のリファレンスである。GLSL ESの詳細は、http://www.khronos.org/registry/gles/specs/2.0/GLSL_ES_Specification_1.0.17.pdfの公式仕様書を参照していただきたい。

▶ データ型

GLSL ESは、JavaScriptとはまったく別のデータ型を揃えている。**表C-1**では、それらのデータ型をまとめてある。

表C-1 データ型

名前	説明

void	関数が戻り値を持たないときや、引数リストが空だということを示すときに使われる特別なデータ型。
	例：
	`void myFunc(void) {`
	`  ...`
	`}`

int	符号付き整数値。
	例：
	`int myInt1 = 14;`
	`int myInt2 = -7;`

float	単精度浮動小数点数。 例： `float myFloat = 3.14159;`
bool	true、falseの論理値。 例： `bool myBool = true;`
vec2、vec3、vec4	それぞれ2、3、4個の浮動小数点数要素を持つベクトル。 例： `vec3 myVec = vec3(1.0, 2.5, 0.25);`

（表C-1 続き）

bvec2、bvec3、bvec4	それぞれ2、3、4個の論理値要素を持つベクトル。 例： `bvec3 myBoolVec = bvec3(true, false, false);`
ivec2、ivec3、ivec4	それぞれ2、3、4個の整数要素を持つベクトル。 例： `ivec3 myIntVec = ivec3(12, -4, 84);`

mat2、mat3、mat4	それぞれ2×2、3×3、4×4の浮動小数点数行列。
	例：
	`mat2 myMat = mat2(`
	`1.0, 0.5,`
	`2.7, 1.5`
	`);`

sampler2D	texture2D関数でアクセスできる2Dテクスチャ。
	例：
	`sampler2D uTexture;`
	`vec4 color = texture2D(uTexture, texCoords);`

samplerCube	textureCube関数でアクセスできるキューブにマッピングされるテクスチャ。
	例：
	`samplerCube uSkyMap;`
	`vec4 color = textureCube(uSkyMap, texCoords);`

▶ 組み込み関数

以下の表では、GLSL ESの組み込み関数を示す。関数は、提供する機能分野に基づいて別々の表に分類されている。

ジェネリックなgenType型で宣言された戻り値と引数は、float、vec2、vec3、vec4型にすることができる。ただし、1つの関数呼び出しで使われるすべてのgenType引数は、みな同じ型でなければならない。戻り値の型は、引数の型によって決まる。

■ 数学関数

表C-2では、GLSL ESがサポートする基本数学関数をまとめてある。

表C-2 組み込み数学関数

関数	戻り値

genType pow(
genType x,
genType n
)
	xのn乗。
	例：
	vec2 result = pow(vec2(2.0, 3.0), vec2(4.0, 5.0));
	// result == vec2(16.0, 243.0)

genType exp(
genType n
) The constant e (2.718...)
	定数*e* (2.718...) のn乗。
	例：
	vec2 result = exp(vec2(2.0, 3.0));
	// result == vec2(7.39, 20.09)

genType exp2(
genType n
)
	2のn乗。
	例：
	vec2 result = exp2(vec2(2.0, 3.0));
	// result == vec2(4.0, 8.0)

genType log(
genType x
)
	xの自然対数。
	例：
	vec2 result = log(vec2(4.0, 8.0));
	// result == vec2(1.39, 2.08)

genType log2(
genType x
)
	xの2を底とする対数。
	例：
	vec2 result = log2(vec2(4.0, 8.0));
	// result == vec2(2.0, 3.0)

genType sqrt(
genType x
)
	xの平方根。
	例：
	vec2 result = sqrt(vec2(9.0, 25.0));
	// result = vec2(3.0, 5.0)

genType inversesqrt(
genType x
)
	xの平方根の逆数、つまり1 / sqrt(x)を計算する。
	例：
	vec2 result = inversesqrt(vec2(9.0, 25.0));
	// result = vec2(0.33..., 0.04)

genType abs(
genType x
)
	xの絶対値。
	例：
	vec2 result = abs(vec2(-4.0, 8.0));
	// result = vec2(4.0, 8.0)

（表C-2続き）

genType sign(
　genType x
)
	xの符号。
	例：
	vec2 result = abs(vec2(-4.0, 8.0));
	// result == vec2(-1.0, 1.0)

genType floor(
　genType x
)
	x以下の最大の整数。
	例：
	vec2 result = floor(vec2(2.7, 14.0));
	// result == vec2(2.0, 14.0)

genType ceil(
　genType x
)
	x以上の最小の整数。
	例：
	vec2 result = ceil(vec2(2.7, 14.0));
	// result == vec2(3.0, 14.0)

genType fract(
　genType x
)
	xの小数部分。
	例：
	vec2 result = fract(vec2(2.7, 14.0));
	// result == vec2(0.7, 0.0)

genType mod(
　genType x,
　genType y
)
genType mod(
　genType x,
　float y
)
	除算 x / y の剰余。
	例：
	vec2 result = mod(vec2(5.0, 19.5), vec2(2.0, 5.0));
	// result == vec2(1.0, 4.5)
	vec2 result = mod(vec2(5.0, 19.5), 2.0);
	// result == vec2(1.0, 1.5)

genType min(
　genType x,
　genType y
)
genType min(
　genType x,
　float y
)
	x、yの最小値。
	例：
	vec2 result = min(vec2(4.0, 15.5), vec2(2.5, 20.0));
	// result == vec2(2.5, 15.0)
	vec2 result = min(vec2(4.0, 15.5), 10.7);
	// result == vec2(4.0, 10.7)

（表C-2続き）

genType max(
genType x,
genType y
)
genType max(
genType x,
float y
)
	x、yの最大値。
	例：
	vec2 result = max(vec2(4.0, 15.5), vec2(2.5, 20.0));
	// result == vec2(4.0, 20.0)
	vec2 result = max(vec2(4.0, 15.5), 10.7);
	// result == vec2(10.7, 15.5)

▶ 三角関数

GLSL ESは、基本的な数学関数のほか、**表C-3**のような三角関数を提供している。

表C-3 組み込み三角関数

関数	戻り値
genType radians(genType degrees)	ラジアンに変換されたdegrees。 radians(degrees) == degrees * π / 180
genType degrees(genType radians)	度に変換されたradians。 degrees(radians) == radians * 180 / π
genType sin(genType angle)	angle（単位ラジアン）の正弦。
genType cos(genType angle)	angle（単位ラジアン）の余弦。
genType tan (genType angle)	angle（単位ラジアン）の正接。
genType asin(genType x)	xの逆正弦。[-π/2, π/2]の範囲の値を返す。

（表C-3続き）

genType acos(genType x)	xの逆余弦。[0, π]の範囲の値を返す。
genType atan(genType r)	rの逆正接。[-π/2, π/2]の範囲の値を返す。
genType atan(genType y, genType x)	y/xの逆正接。[-π, π]の範囲の値を返す。

▶ ベクトル、行列関数

表C-4には、ベクトルや行列を操作しやすくする関数をまとめてある。

表C-4 組み込みのベクトル、行列関数

関数	戻り値
float length(genType v)	ベクトルvの長さ。 length(v) == sqrt(v.x*v.x + v.y*v.y + ...)
float distance(genType p0, genType p1)	点p0、p1の距離。 distance(p0, p1) == length(p1 - p0)
float dot(genType v0, genType v1)	ベクトルv0、v1のドット積。 dot(v0, v1) == v0.x*v1.x + v0.y+v1.y + ...
vec3 cross(vec3 v0, vec3 v1)	ベクトルv0、v1のクロス積。 cross(v0, v1) == vec3(v0.y * v1.z - v1.y * v0.z, v0.z * v1.x - v1.z * v0.x, v0.x * v1.y - v1.x * v0.y)

（表C-4続き）

genType normalize(genType v)	単位長に正規化されたベクトルv。 normalize(v) == v / length(v)

genType faceforward(
genType N,
genType I,
genType Nref
)
	サーフェス法線Nrefが生成ベクトルIと反対向きになっているなら、ベクトルNを反転する。
	if (dot(Nref, I) < 0)
	return N
	else
	return -N

genType reflect(genType I, genType N)	生成ベクトルIが法線Nを持つサーフェスで反射したもの。 reflect(I, N) == I - 2 * dot(N, I) * N

genType refract(
genType I,
genType N,
float eta
)
	生成ベクトルIが屈折率etaで法線Nを持つサーフェスに対して起こす屈折のベクトル。
	屈折ベクトルは次のようにして計算する。
	k = 1.0 - eta * eta * (1.0 - dot(N, I) * dot(N, I))
	if (k < 0.0)
	return genType(0.0)
	else
	return eta * I - (eta * dot(N, I) + sqrt(k)) * N

mat matrixCompMult(mat m0, mat m1)	行列m0、m1の要素ごとの乗算。ここで、matはmat2、mat3、mat4のいずれかである。

ベクトルの関係演算

<、>、==、!=などの標準の関係演算子は、すべての要素をまとめて見なければベクトルを比較できない。**表C-5**には、ベクトルの要素ごとの比較、評価を行う関数をまとめてある。

表C-5 組み込みのベクトル関係関数

関数	戻り値
bvec lessThan(vec v0, vec v1)	v0 < v1の要素ごとの評価。 lessThan(vec2(1.0, 2.0), vec2(4.0, 2.0)) == bvec2(true, false)
bvec lessThanEqual(vec v0, vec v1)	v0 <= v1の要素ごとの評価。 lessThanEqual(vec2(1.0, 2.0), vec2(4.0, 2.0)) == bvec2(true, true)
bvec greaterThan(vec v0, vec v1)	v0 > v1の要素ごとの評価。 greaterThan(vec2(1.0, 2.0), vec2(4.0, 2.0)) == bvec2(false, false)
bvec greaterThanEqual(vec v0, vec v1)	v0 >= v1の要素ごとの評価。 greaterThanEqual(vec2(1.0, 2.0), vec2(4.0, 2.0)) == bvec2(false, true)
bvec equal(vec v0, vec v1) bvec equal(bvec v0, bvec v1)	v0 == v1の要素ごとの評価。 equal(vec2(1.0, 2.0), vec2(4.0, 2.0)) == bvec2(false, true) equal(bvec2(true, false), bvec2(false, false)) == bvec2(false, true)
bvec notEqual(vec v0, vec v1) bvec notEqual(bvec v0, bvec v1)	v0 != v1の要素ごとの評価。 notEqual(vec2(1.0, 2.0), vec2(4.0, 2.0)) == bvec2(true, false) notEqual(bvec2(true, false), bvec2(false, false)) == bvec2(true, false)

（表C-5続き）

bool any(bvec v)	ベクトルvの任意の要素がtrueならtrue。 any(bvec3(false, false, false)) == true any(bvec3(false, true, false)) == true
bool all(bvec v)	ベクトルvのすべての要素がtrueならtrue。 all(bvec3(false, true, false)) == false all(bvec3(true, true, true)) == true
bvec not(bvec v)	!vの要素ごとの評価。 not(bvec2(true, false)) == bvec2(false, true)

▶ ヘルパー関数

表C-6には、範囲制限やブレンディングなどの処理で役に立つさまざまな関数をまとめてある。

表C-6 組み込みヘルパー関数

関数	戻り値
genType clamp(genType x, genType minVal, genType maxVal) genType clamp(genType x, float minVal, float maxVal)	xの値を[minVal,maxVal]の範囲、つまりmax(minVal, min(maxVal, x))に制限する。 clamp(vec2(0.5, 1.7), vec2(1.0, 1.2), vec2(1.3, 1.5)) == vec2(1.0, 1.5) clamp(vec2(0.5, 1.7), 0.7, 1.3) == vec2(0.7, 1.3)

（表C-6続き）

genType mix(　genType x, 　genType y, 　genType a) genType mix(　genType x, 　genType y, 　float a)	xとyの間の線形補間。 mix(x, y, a) == x * (1.0 - a) + y * a
genType step(　genType edge, 　genType x) genType step(　float edge, 　genType x)	xの要素のうち、edgeよりも小さいものは0.0、そうでないものは1.0にする。
genType smoothstep(　genType edge0, 　genType edge1, 　genType x) genType smoothstep(　float edge0, 　float edge1, 　genType x)	xの要素のうち、edge0以下のものは0.0、edge1以上のものは1.0にする。xがedge0とedge1の間にあるときには、3次エルミート補間を返す。 t = clamp((x - edge0) / (edge1 - edge0), 0.0, 1.0) return t * t * (3 - 2 * t)

▶ 組み込み変数と定数

GLSL ESは、組み込み関数のほか、いくつかの定数と変数を公開している。**表C-7**には、頂点、フラグメントシェーダから読み出せる定数をまとめてある。これらの定数の値は、OpenGL ESの実装によって異なる。

表C-7 定数

定数	最小値	説明
`int gl_MaxVertexAttribs`	8	頂点属性の最大個数。
`int gl_MaxVertexUniformVectors`	128	Uniform頂点ベクトルの最大個数。
`int gl_MaxFragmentUniformVectors`	16	Uniformフラグメントベクトルの最大個数。
`int gl_MaxVaryingVectors`	8	Varyingベクトルの最大個数。
`int gl_MaxVertexTextureImageUnits`	0	頂点シェーダ内のテクスチャイメージユニットの最大個数。
`int gl_MaxCombinedTextureImageUnits`	8	頂点シェーダとフラグメントシェーダ全体でのテクスチャイメージユニットの最大個数。
`int gl_MaxTextureImageUnits`	8	テクスチャユニットの最大個数。
`int gl_MaxDrawBuffers`	1	フラグメントシェーダ内の`gl_FragData`配列に含まれる出力カラーの最大個数。

表C-8には、頂点シェーダに固有で、そのためフラグメントシェーダでは使えない変数をまとめてある。

表C-8 頂点シェーダ変数

変数	説明
`vec4 gl_Position`	頂点シェーダが計算した現在の頂点の変換、射影された位置。
`float gl_PointSize`	`gl.POINTS`タイプのジオメトリを描画するときに、点スプライトのサイズを指定するために使われる。デフォルトの値は1.0。

表C-9には、フラグメントシェーダに固有な変数をまとめてある。これらの変数は、頂点シェーダでは使えない。

表C-9 フラグメントシェーダ変数

変数	説明
`vec4 gl_FragColor`	現在のフラグメントに与えられたカラー値。
`vec4 gl_FragCoord`	現在のフラグメントのウィンドウからの相対座標。
`vec4 gl_FragData [gl_MaxDrawBuffers]`	現在のフラグメントのフラグメントデータの配列。フラグメントシェーダは、`gl_FragCoord`、`gl_FragData`のどちらにも値を代入してはならない。
`vec2 gl_PointCoord`	ポイントスプライトを描くときに、`gl_PointCoord`は、フラグメントのポイントからの2次元相対座標を保持する。座標は、0.0から1.0までである。
`bool gl_FrontFacing`	フラグメントが前面に出ているプリミティブの一部かどうかを示す論理値。