

**PlayStation®Edge**  
**オフラインツール用ジオメトリ**  
**ライブラリ リファレンス**

© 2010 Sony Computer Entertainment Inc.  
All Rights Reserved.  
SCE Confidential

# 目次

はじめに .....	4
このドキュメントについて .....	5
ツール用のデータ型 .....	7
EdgeGeomSegmentFormat .....	8
EdgeGeomScene .....	10
EdgeGeomSegment .....	13
EdgeGeomKCacheOptimizerUserData .....	16
EdgeGeomKCacheOptimizerHillclimberUserData .....	17
EdgeGeomPartitionerInput .....	18
EdgeGeomPartitionElementCounts .....	20
EdgeGeomPartitionerOutput .....	21
EdgeGeomSpuVertexAttributeDefinition .....	22
EdgeGeomSpuVertexFormat .....	24
EdgeGeomRsxVertexAttributeDefinition .....	25
EdgeGeomRsxVertexFormat .....	26
libedgegeomtool関数 .....	27
edgeGeomPartitioner .....	28
edgeGeomAlloc .....	29
edgeGeomFree .....	30
edgeGeomSetAllocFunc .....	31
edgeGeomSetFreeFunc .....	32
edgeGeomGetCommandBufferHoleSize .....	33
edgeGeomGetScratchBufferSizeInQwords .....	34
edgeGeomGetSpuVertexFormat .....	35
edgeGeomGetRsxVertexFormat .....	36
edgeGeomGetSpuVertexFormatId .....	37
edgeGeomGetRsxVertexFormatId .....	38
edgeGeomSpuVertexFormatIsValid .....	39
edgeGeomRsxVertexFormatIsValid .....	40
edgeGeomGetSpuVertexAttributeSize .....	41
edgeGeomGetRsxVertexAttributeSize .....	42
edgeGeomGetAttributeSlotIndex .....	43
edgeGeomSetAttributeSlotIndex .....	44
edgeGeomComputeTriangleCentroids .....	45
edgeGeomGetBlendedVertexes .....	46
edgeGeomBlendShapeAffectsSegment .....	48
edgeGeomMakeSpuConfigInfo .....	50
edgeGeomMakeIndexBuffer .....	52
edgeGeomMakeSpuVertexBuffer .....	53
edgeGeomMakeRsxVertexBuffer .....	55
edgeGeomMakeSkinningBuffer .....	57
edgeGeomMakeBlendShapeBuffer .....	59
edgeGeomMergeIdenticalVertexes .....	61
edgeGeomPartitionSceneIntoSegments .....	62

---

edgeGeomFreeSegmentData .....	64
edgeGeomTriangulatePolygons .....	65
edgeGeomMakeSpuStreamDescription .....	67
edgeGeomMakeRsxStreamDescription .....	68
edgeGeomKCacheOptimizer .....	69
edgeGeomKCacheOptimizerHillclimber .....	71
<b>コールバック関数 .....</b>	<b>72</b>
EdgeGeomVertexCacheOptimizerFunc .....	73
EdgeGeomCustomPartitionDataSizeFunc .....	74
<b>列挙 .....</b>	<b>75</b>
EdgeGeomIndexesFlavor .....	76
EdgeGeomSkinningFlavor .....	77
EdgeGeomMatrixFormat .....	78
EdgeGeomCullingFlavor .....	79

# はじめに

---

# このドキュメントについて

---

## 目的

このドキュメントは、Edge ライブラリのオフラインツール用ジオメトリコンポーネントである libedgegeomtool の API リファレンスです。このコンポーネントをアセットパイプラインに組込むと、データを効率的に管理し、実行時に高いパフォーマンスを維持することができます。

Edge ジオメトリコンパイラツール (edgegeomcompiler) は、libedgegeomtool をアセットパイプラインの中でどのように利用できるのかを示すサンプルツールです。具体的には、COLLADA™ フォーマットのデータを取り込んで Edge ライブラリに渡し、ランタイムサンプルにロード可能なバイナリファイルを生成する方法を示しています。FCollada を使用しているため、edgeanimcompiler は、現時点では Windows でしかサポートされていません。

## 対象読者と前提条件

このドキュメントは、PlayStation®3 用の高性能アプリケーションを書こうとしている PlayStation®3 デベロッパーのため書かれたもので、以下の点を熟知していることを前提にしています。

- C と C++
- PlayStation®3 のハードウェア
- SCE の標準ライブラリ関数

## 関連ドキュメント

このリファレンスと以下のドキュメントを併用することにより、Edge ライブラリの使用法やリファレンスについての完全な情報を得ることができます。

- 「PlayStation® Edge ライブラリ 概要」
- 「PlayStation® Edge ジオメトリライブラリ クイックスタート」
- 「PlayStation® Edge ジオメトリライブラリ リファレンス」
- 「PlayStation® Edge アニメーションライブラリ リファレンス」
- 「PlayStation® Edge オフラインツール用アニメーションライブラリ リファレンス」
- 「PlayStation® Edge Zlib ライブラリ リファレンス」
- 「PlayStation® Edge LZMA ライブラリ リファレンス」
- 「PlayStation® Edge LZMA ライブラリ リファレンス」
- 「PlayStation® Edge DXT ライブラリ リファレンス」
- 「PlayStation® Edge Post ライブラリ リファレンス」

## 表記法

このドキュメントでは、以下のような印刷上の表記法を使います。

記法	意味
等幅フォント	プログラミングコード、および処理命令、レジスタ名、データ型、イベント、ファイル名などのリテラルを表します。また、関数、構造体、マクロ名なども表します。
等幅フォント+太字	構造体や関数の定義の中でのみ、構造体や関数の名前を示します。
等幅フォント+斜体	引数、パラメータ、変数を表します。
<u>青字 + 下線</u>	ハイパーリンクを表します（青色で表示されるのは、カラープリンタもしくはオンラインの場合だけです）。

## ツール用のデータ型

# EdgeGeomSegmentFormat

Edge ツールによって生成されたセグメントのフォーマットに関する全データを格納するためのコンテナ構造体

## 定 義

```
#include <libedgegeomtool_wrap.h>
struct EdgeGeomSegmentFormat
{
    EdgeGeomSpuVertexFormat *m_spuInputVertexFormats[2];
    EdgeGeomSpuVertexFormat *m_spuInputVertexDeltaFormat;
    EdgeGeomRsxVertexFormat *m_spuOutputVertexFormat;
    EdgeGeomRsxVertexFormat *m_rsxOnlyVertexFormat;
    EdgeGeomSkinningFlavor m_skinType
    EdgeGeomIndexesFlavor m_indexesType;
    EdgeGeomMatrixFormat m_skinMatrixFormat;
};
```

## メ ン バ

<i>m_spuInputVertexFormats</i>	2つのSPU入力頂点ストリームの頂点フォーマットへのポインタ。1番目のエントリは常に存在していなければなりません。2番目のエントリは、SPU入力ストリームが1つだけの場合はNULLでもかまいません。
<i>m_spuInputVertexDeltaFormat</i>	ブレンド形状頂点デルタストリームによって使用される頂点フォーマットへのポインタ。シーンにブレンド形状データが含まれていない場合はNULLでもかまいません。
<i>m_spuOutputVertexFormat</i>	SPU出力頂点ストリームの頂点フォーマットへのポインタ。NULL以外である必要があります。
<i>m_rsxOnlyVertexFormat</i>	セグメントのRSX®専用頂点ストリームの、頂点フォーマットへのポインタ。このストリームには、SPUによる処理の行われない属性が含まれている必要があります。
<i>m_skinType</i>	シーンで使われるスキニングアルゴリズムを定義します。最高の効率を得るには、シーンデータに正しく適用できるアルゴリズムの中で最も簡単なものを選択してください。
<i>m_indexesType</i>	実行時の入力インデックス・三角形ストリームのフォーマットを決めます。
<i>m_skinMatrixFormat</i>	スキニングが有効になっている場合に、スキニング行列のフォーマットを決めます。

## 解 説

この構造体は、生のジオメトリデータ ([EdgeGeomScene](#) オブジェクトなど) を、Edgeジオメトリランタイムで処理するのに適したフォーマットに変換するために必要な情報のすべてをまとめたものです。  
*m\_indexesType* には、以下の値を設定することができます。

列挙型	値	解説
<i>KindexesU16TriangleListCW</i>	0	16ビットのインデックス化されたトライアングルリスト。順序は時計回り。
<i>kIndexesU16TriangleListCCW</i>	1	16ビットのインデックス化されたトライアングルリスト。順序は反時計回り。



列挙型	値	解説
kIndexesCompressedTriangleListCW	2	圧縮されたインデックス化されたトライアングルリスト。順序は時計回り。(SPUのみ)
kIndexesCompressedTriangleListCCW	3	圧縮されたインデックス化されたトライアングルリスト。順序は反時計回り。(SPUのみ)

`m_skinMatrixFormat` には、以下の値を設定することができます。

列挙型	値	解説
kMatrix3x4RowMajor	0	これは Edge のネイティブの行列型で、メモリの使用量に関して最も効率的です。この行列は、「DirectX スタイル」の 4x4 行列の上 3 行です。4 行目は常に [0, 0, 0, 1] なので、明示的に格納する必要はありません。
kMatrix4x4RowMajor	1	「DirectX スタイル」の行優先形式の 4x4 行列を指定します。この型が提供されるのは、主に便宜上の理由からです。4x4 行列は、3x4 行列より 33% 多くのメモリを使用することは言うまでもありません。
kMatrix4x4ColumnMajor	2	「OpenGL スタイル」の列優先形式の 4x4 行列を指定します。この型が提供されるのは、主に便宜上の理由からです。4x4 行列は、3x4 行列より 33% 多くのメモリを使用することは言うまでもありません。

この構造体に関連するメモリ管理は、すべて呼出し側の責任となります。

## 関 連 項 目

[EdgeGeomScene](#)、[EdgeGeomSegment](#)

# EdgeGeomScene

Edge ジオメトリツールでロードされた入力シーンに関連する全データを格納するコンテナ構造体で、分割されて実行時セグメントデータに変換されます

## 定 義

```
#include <libedgegeomtool_wrap.h>
struct EdgeGeomScene
{
    // 三角形データ
    uint32_t m_numTriangles;
    uint32_t *m_triangles;
    int32_t *m_materialIdPerTriangle;

    //メイン頂点データ
    uint32_t m_numVertexes;
    uint32_t m_numFloatsPerVertex;
    float m_vertexes;
    uint8_t m_numVertexAttributes;
    uint16_t *m_vertexAttributeIndexes;
    EdgeGeomAttributeId *m_vertexAttributeIds;

    //ブレンド形状データ
    uint32_t m_numBlendShapes;
    uint32_t m_numFloatsPerDelta;
    float *m_vertexDeltas;
    uint8_t m_numBlendedAttributes;
    uint16_t *m_blendedAttributeIndexes;
    EdgeGeomAttributeId *m_blendedAttributeIds;

    //スキニングデータ
    int32_t *m_matrixIndexesPerVertex;
    float *m_skinningWeightsPerVertex;
};
```

## メ ン バ

<code>m_numTriangles</code>	シーン中の三角形の数
<code>m_triangles</code>	シーン中の三角形のデータ。1つの三角形が3つの32ビット整数で表されます。この配列の要素は、最低でも <code>m_numTriangles*3</code> 個以上である必要があります。
<code>m_materialIdPerTriangle</code>	シーン中の各三角形用のマテリアル ID の配列。マテリアル ID は、マテリアル全体で一意であれば、シェーダへのポインタ、シェーダ名のハッシュ、単純な整数インデックスなど、なんでもかまいません。この配列の要素は、最低でも <code>m_numTriangles</code> 個以上である必要があります。
<code>m_numVertexes</code>	シーン中の頂点の数。
<code>m_numFloatsPerVertex</code>	<code>m_vertexes</code> 配列中の各頂点のストライド (32 ビット float)。
<code>m_vertexes</code>	各頂点が連続するデータブロックになるようにインタリーブされている頂点属性ストリームとして表されたシーン頂点データ。この属性はすべて、32 ビット精度で表されます。この配列の要素は、最低でも $(m\_numVertexes * m\_numFloatsPerVertex)$ 個以上である必要があります。
<code>m_numVertexAttributes</code>	一次頂点ストリーム中の各頂点の頂点属性の数。

<code>m_vertexAttributeIndexes</code>	<code>m_vertexes</code> 中の各頂点に対応する float データのブロックの中の、全頂点属性インデックスの配列。この配列には、最低でも <code>m_numVertexAttributes</code> 個の要素が含まれている必要があり、また要素の順序は、 <code>m_vertexAttributeIds</code> 配列の要素と同じでなくてはなりません。
<code>m_vertexAttributeIds</code>	全頂点属性の属性 ID の配列。この配列には、最低でも <code>m_numVertexAttributes</code> 個の要素が含まれている必要があり、また要素の順序は、 <code>m_vertexAttributeIndexes</code> 配列の要素と同じでなくてはなりません。
<code>m_numBlendShapes</code>	シーン中のブレンド形状の数。ゼロの場合、以下のブレンド形状関連の値や配列は、未定義のままでもかまいません。
<code>m_numFloatsPerDelta</code>	<code>m_vertexDeltas</code> 配列中の各頂点デルタのストライド (32 ビット float)。
<code>m_vertexDeltas</code>	各ブレンド形状が連続するデータブロックになるようにインタリーブされている頂点属性ストリームとして表されたシーンのブレンド形状データ。シーンの全頂点のデルタは、各ブレンド形状ごとに連続するメモリブロック (1 形状当たり $m\_numVertexes * m\_numFloatsPerDelta$ 個エントリの要素) として提供されます。この属性はすべて、32 ビット精度で表されます。この配列の要素は、最低でも $(m\_numBlendShapes * m\_numVertexes * m\_numFloatsPerDelta)$ 個以上である必要があります。
<code>m_numBlendedAttributes</code>	ブレンド形状ストリーム中の各頂点デルタの頂点属性の数。メインシーンの全属性がブレンドの対象になるとは限りませんが、デルタストリーム中の各属性はメイン頂点ストリーム中に存在する「必要」があります。
<code>m_blendedAttributeIndexes</code>	<code>m_vertexDeltas</code> 中の各頂点デルタに対応する float データのブロックの中の、全ブレンド頂点属性インデックスの配列。この配列には、最低でも <code>m_numBlendedAttributes</code> 個の要素が含まれている必要があり、また要素の順序は、 <code>m_blendedAttributeIds</code> 配列の要素と同じでなくてはなりません。
<code>m_blendedAttributeIds</code>	全ブレンド頂点属性の属性 ID の配列。この配列には、最低でも <code>m_numBlendedAttributes</code> 個の要素が含まれている必要があり、また要素の順序は、 <code>m_blendedAttributeIndexes</code> 配列の要素と同じでなくてはなりません。
<code>m_matrixIndexesPerVertex</code>	この頂点に影響する 4 つのスキニングボーン行列のシーンボーン配列へのインデックスが含まれています。頂点のボーンインフルエンスが 4 つ未満である場合、残りのインデックスは -1 に設定する必要があります。この配列は、 $m\_numVertexes * 4$ 個の要素をもつ必要があります。
<code>m_skinningWeightsPerVertex</code>	各頂点の 4 つのスキニングボーンの重みが含まれています。各重みは 0.0~1.0 の範囲にある必要があります。まだ特定の頂点の 4 つの重みの和はちょうど 1.0 である必要があります。使われていないボーンの重みは、0.0 に設定する必要があります。この配列は、 $m\_numVertexes * 4$ 個の要素をもつ必要があります。

## 解 説

この構造体には、Edge ツールへの入力シーンのジオメトリデータがまとめられており、その中には、三角形、頂点、ブレンド形状、スキニングデータなどが含まれています。この構造体に関連するメモリ管理は、すべてユーザ側の責任となります。

以下は、三角形 3 つとマテリアル 2 つを示す、三角形データの正しいフォーマットの例です。

*m\_triangles*:

0	1	2	2	1	3	3	2	4
---	---	---	---	---	---	---	---	---

*m\_materialIdPerTriangle*:

0	0	1
---	---	---

*m\_numTriangles*:

3
---

*m\_vertexes* の記憶フォーマットは以下の通りです。この中の同じ番号は同じ頂点を表し、各頂点の中の細いボックスは頂点の属性を表します。

0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4	5	5	5	5	.
																								.
																								.

*m\_vertexDeltas* の保存形式も、*m\_vertexAttributes* に似ていますが、この配列では、各ブレンド形状の頂点データのすべてがまとまって存在しており、1 頂点当たりの要素の数もベース頂点属性配列とは違います。

0	0	0	1	1	1	2	2	2	3	3	3	4	4	4	5	5	5	.
																		.
																		.
0	0	0	1	1	1	2	2	2	3	3	3	4	4	4	5	5	5	.
																		.
																		.
0	0	0	1	1	1	2	2	2	3	3	3	4	4	4	5	5	5	.
																		.
																		.

## 関 連 項 目

# EdgeGeomSegment

Edge ツールによって生成された最終的な実行時データすべてを PlayStation®3 ネイティブのビッグエンディアン形式で格納し、そのままディスクに書き込むことのできるコンテナ構造体

## 定 義

```
#include <libedgegeomtool_wrap.h>
struct EdgeGeomSegment
{
    uint32_t m_materialId;

    uint8_t *m_spuConfigInfo;

    uint8_t *m_indexes;
    uint16_t m_indexesSizes[2];

    uint8_t *m_spuVertexes[2];
    uint16_t m_spuVertexesSizes[6];
    uint32_t m_fixedOffsetsSizes[2];
    uint32_t *m_fixedOffsetPtrs[2];

    uint8_t *m_rsxOnlyVertexes;
    uint32_t m_rsxOnlyVertexesSize;

    uint8_t *m_spuInputStreamDescriptions[2];
    uint16_t m_spuInputStreamDescriptionSizes[2];
    uint8_t *m_spuOutputStreamDescription;
    uint16_t *m_spuOutputStreamDescriptionSize;
    uint8_t *m_rsxOnlyStreamDescription;
    uint16_t *m_rsxOnlyStreamDescriptionSize;

    uint16_t m_skinMatricesByteOffsets[2];
    uint16_t m_skinMatricesSizes[2];
    uint8_t *m_skinIndexesAndWeights;
    uint16_t m_skinIndexesAndWeightsSizes[2];

    uint32_t m_ioBufferSize;
    uint32_t m_scratchSize;

    uint32_t m_numBlendShapes;
    uint16_t *m_blendShapeSizes;
    uint8_t **m_blendShapes;
};
```

## メ ン バ

*m\_materialId*

このセグメント中の全三角形のマテリアルID。この値は、[EdgeGeomScene](#) オブジェクトの一部として渡される

*m\_spuConfigInfo*

*m\_materialIdPerTriangle* 配列から抽出されます。

*m\_indexes*

セグメントの *EdgeGeomSpuConfigInfo* 構造体へのポインタ  
セグメントのインデックスバッファへのポインタ

*m\_indexesSizes*

インデックスバッファの DMA タグサイズ

*m\_spuVertexes*

セグメントの一次および二次 SPU 入力頂点バッファへのポインタ。必要とされる SPU 頂点ストリームが1つだけの場合、二番目の要素は NULL でもかまいません。

<code>m_spuVertexesSizes</code>	SPU 入力頂点バッファの DMA タグサイズ (頂点ストリーム当たり 3 要素)。使われていない要素はゼロに設定する必要があります。
<code>m_fixedOffsetsSizes</code>	一次および二次固定小数点オフセットテーブルのサイズ (バイト)。使われていない場合には、ゼロに設定します。
<code>m_fixedOffsetPtrs</code>	2 つの入力頂点ストリームの固定小数点オフセットテーブルそれぞれへのポインタ。これらの値は、ストリームを元の範囲に戻すために加算される固定小数点値です。使われていない場合には、NULL に設定します。
<code>m_rsxOnlyVertexes</code>	このセグメントのオプションの RSX®専用頂点ストリームへのポインタ。これらの属性は SPU を介して送信されることはありません。未使用の場合は NULL になります。
<code>m_rsxOnlyVertexesSize</code>	RSX®専用の頂点ストリームのサイズ (バイト)。未使用の場合はゼロになります。
<code>m_spuInputStreamDescriptions</code>	2 つの SPU 入力頂点ストリーム記述へのポインタ。使用する頂点ストリームが 1 つだけの場合、二番目の要素は NULL でもかまいません。
<code>m_spuInputStreamDescriptionSizes</code>	2 つの SPU 入力頂点ストリーム記述のサイズ (バイト)
<code>m_spuOutputStreamDescription</code>	SPU 出力頂点ストリーム記述へのポインタ
<code>m_spuOutputStreamDescriptionSize</code>	SPU 出力頂点ストリーム記述のサイズ (バイト)
<code>m_rsxOnlyStreamDescription</code>	オプションの RSX®専用頂点ストリームのストリーム記述へのポインタ。このセグメントに RSX®専用頂点ストリームが含まれている場合に限り NULL 以外の値になります。
<code>m_rsxOnlyStreamDescriptionSize</code>	RSX®専用ストリーム記述のサイズ (バイト)。未使用の場合はゼロになります。
<code>m_skinMatricesByteOffsets</code>	このセグメントによって使われる 2 つのスキニング行列範囲の先頭へのバイトオフセット
<code>m_skinMatricesSizes</code>	このセグメントによって使われる 2 つのスキニング行列範囲のサイズ (バイト)
<code>m_skinIndexesAndWeights</code>	このセグメントのスキニングインデックス/重みバッファへのポインタ
<code>m_skinIndexesAndWeightsSizes</code>	スキニングインデックス/重み DMA タグのサイズ (バイト)。
<code>m_ioBufferSize</code>	このセグメントの SPU 上での IO バッファの最大使用量のサイズ (バイト)
<code>m_scratchSize</code>	このセグメントの SPU 上でのスクラッチバッファの最大使用量のサイズ (16 バイトのクワッドワード単位)。
<code>m_numBlendShapes</code>	セグメントに含まれるブレンド形状の数。この数は、このセグメントに潜在的に影響を与える可能性のあるブレンド形状の数で、通常、シーン中のブレンド形状の合計より少ないです。
<code>m_blendShapeSizes</code>	このセグメントのブレンド形状バッファのサイズ (バイト) の配列。この配列には、 <code>m_numBlendShapes</code> 個の要素があります。
<code>m_blendShapes</code>	このセグメントのブレンド形状バッファへのポインタの配列。この配列には、 <code>m_numBlendShapes</code> 個の要素があります。

## 解 説

この構造体には、特定のジオメトリセグメントのデータのすべてが含まれています。この構造体のメンババッファは、必要に応じてビッグエンディアンにバイトスワップされ、そのままディスクに書き込み

る（libedgegeomtool が PPU 上で実行されている場合には、すぐに処理できる）状態になっています。

ジオメトリセグメントを作成する最も簡単な方法は、[edgeGeomPartitionSceneIntoSegments\(\)](#) 関数を使う方法です。また、この構造体のメンバは、以下のlibedgegeomtoolの関数で生成することもできます。

<a href="#">edgeGeomMakeSpuConfigInfo()</a>	<i>m_spuConfigInfo</i>
<a href="#">edgeGeomMakeIndexBuffer()</a>	<i>m_indexes</i>
	<i>m_indexesSizes</i>
<a href="#">edgeGeomMakeSpuVertexBuffer()</a>	<i>m_spuVertexes</i>
	<i>m_spuVertexesSizes</i>
	<i>m_fixedOffsetsSizes</i>
	<i>m_fixedOffsetPtrs</i>
<a href="#">edgeGeomMakeRsxVertexBuffer()</a>	<i>m_rsxOnlyVertexes</i>
	<i>m_rsxOnlyVertexesSize</i>
<a href="#">edgeGeomMakeSpuStreamDescription()</a>	<i>m_spuInputStreamDescriptions</i>
	<i>m_spuInputStreamDescriptionSizes</i>
<a href="#">edgeGeomMakeRsxStreamDescription()</a>	<i>m_spuOutputStreamDescription</i>
	<i>m_spuOutputStreamDescriptionSize</i>
	<i>m_rsxOnlyStreamDescription</i>
	<i>m_rsxOnlyStreamDescriptionSize</i>
<a href="#">edgeGeomMakeSkinningBuffer()</a>	<i>m_skinMatricesByteOffsets</i>
	<i>m_skinMatricesSizes</i>
	<i>m_skinIndexesAndWeightsSizes</i>
	<i>m_skinIndexesAndWeights</i>
<a href="#">edgeGeomPartitioner()</a>	<i>m_ioBufferSize</i>
<a href="#">edgeGeomGetScratchBufferSizeInQwords()</a>	<i>m_scratchSize</i>
<a href="#">edgeGeomMakeBlendShapeBuffer()</a>	<i>m_numBlendShapes</i>
	<i>m_blendShapeSizes</i>
	<i>m_blendShapes</i>

この構造体に関連するメモリ管理は、すべて呼出し側の責任となります。

## 関 連 項 目

[edgeGeomPartitionSceneIntoSegments](#)

# EdgeGeomKCacheOptimizerUserData

[edgeGeomKCacheOptimizer\(\)](#) 関数のアルゴリズム専用のデータコンテナ

## 定 義

```
#include <libedgegeomtool.h>
struct EdgeGeomKCacheOptimizerUserData
{
    float m_fifoMissCost,
    float m_lruMissCost,
    float m_k1,
    float m_k2,
    float m_k3
};
```

## メ ン バ

<code>m_fifoMissCost</code>	変換後のキャッシュミス 1 回当たりのコスト係数。頂点プログラムへの入力属性の数にすることをお勧めします。
<code>m_lruMissCost</code>	ミニキャッシュミス 1 回当たりのコスト係数。頂点プログラムからの出力属性の数の 4 倍にすることをお勧めします。
<code>m_k1</code>	特定の頂点近傍におけるキャッシュミスの総コストの重要度を制御する係数。1.0 を推奨。
<code>m_k2</code>	特定の頂点近傍におけるレンダリングされる面数の重要度を制御する係数。0.25 を推奨。
<code>m_k3</code>	変換後のキャッシュにおける中心頂点の最終的な年齢の重要度を制御している係数。0.8 を推奨。

## 解 説

libedgegeomtool では、ユーザが性能特性（正確さと実行時間の間のトレードオフなど）が最もニーズに合うような実装を選択できるように、頂点キャッシュ最適化機能の標準インタフェースをサポートしています。さまざまな実装をサポートしつつ統一されたインタフェースを維持するために、頂点キャッシュオブティマイザ関数は、それぞれ関数固有の任意のユーザデータへのポインタを受け取れるようになっています。

この構造体には、[edgeGeomKCacheOptimizer\(\)](#) 関数用のユーザデータが含まれています。

## 関 連 項 目

[edgeGeomKCacheOptimizer](#)、[EdgeGeomPartitionerInput](#)



# EdgeGeomKCacheOptimizerHillclimberUserData

[edgeGeomKCacheOptimizer\(\)](#) 関数のアルゴリズム専用のデータコンテナ

## 定 義

```
#include <libedgegeomtool.h>
struct EdgeGeomKCacheOptimizerHillclimberUserData
{
    uint32_t m_numIterations,
    EdgeGeomIndexesFlavor m_indexesType,
    uint32_t m_numRsxInputAttributes,
    uint32_t m_numRsxOutputAttributes
};
```

## メ ン バ

<code>m_numIterations</code>	実行する山登り法の繰返しの数。0 の場合、kcache オプティマイザは一度だけデフォルトパラメータで実行されます。この繰返しの値を 100 以上に設定すると、処理が極めて遅くなり、また、目立った結果の改善にもつながりません。
<code>m_indexesType</code>	出力三角形リストのフォーマット。スコア計算の前に三角形リストが圧縮されているかを判定し、最適化された三角形を回転させて、最高の圧縮率を実現するために使われます。
<code>m_numRsxInputAttributes</code>	頂点プログラムへの入力として渡された頂点属性の数。
<code>m_numRsxOutputAttributes</code>	頂点プログラムからの出力として渡された属性の数。

## 解 説

libedgegeomtool では、ユーザが性能特性（正確さと実行時間の間のトレードオフなど）が最もニーズに合うような実装を選択できるように、頂点キャッシュ最適化機能の標準インタフェースをサポートしています。さまざまな実装をサポートしつつ統一されたインタフェースを維持するために、頂点キャッシュオプティマイザ関数は、それぞれ関数固有の任意のユーザデータへのポインタを受け取れるようになっています。

この構造体には、[edgeGeomKCacheOptimizerHillclimber\(\)](#) 関数用のユーザデータが含まれています。

## 関 連 項 目

[edgeGeomKCacheOptimizerHillclimber](#)、[EdgeGeomPartitionerInput](#)

# EdgeGeomPartitionerInput

[edgeGeomPartitioner\(\)](#) に入力パラメータを指定する構造体

## 定 義

```
#include <libedgegeomtool_partitioner.h>
struct EdgeGeomPartitionerInput
{
    uint32_t m_numTriangles;
    uint32_t *m_triangleList;
    uint32_t m_numInputAttributes;
    uint32_t m_numOutputAttributes;
    uint32_t m_inputVertexStride[2];
    uint32_t m_outputVertexStride;
    EdgeGeomSkinningFlavor m_skinningFlavor;
    EdgeGeomIndexesFlavor m_indexListFlavor;
    EdgeGeomMatrixFormat m_skinningMatrixFormat;

    EdgeGeomVertexCacheOptimizerFunc m_cacheOptimizerFunc;
    void *m_cacheOptimizerUserData;
    EdgeGeomCustomPartitionDataSizeFunc m_customDataSizeFunc;
    float *m_triangleCentroids;
    int32_t *m_skinningMatrixIndexesPerVertex;
    uint32_t m_deltaStreamVertexStride;
    uint32_t *m_blendedVertexIndexes;
    uint32_t m_numBlendedVertexes;
};
```

## メ ン バ

<i>m_numTriangles</i>	三角形の数
<i>m_triangleList</i>	頂点インデックスの配列。連続する 3 つのインデックスが三角形を表します。
<i>m_numInputAttributes</i>	(一次および二次入力ストリーム中の) SPU 入力頂点属性数の合計
<i>m_numOutputAttributes</i>	(SPU から GPU に送信される) 出力属性の数。
<i>m_inputVertexStride</i>	2 つの SPU 入力頂点ストリーム中の頂点のサイズ (バイト)
<i>m_outputVertexStride</i>	SPU 出力頂点ストリーム中の頂点のサイズ (バイト)
<i>m_skinningFlavor</i>	実行されるスキニングの種類
<i>m_indexListFlavor</i>	入力インデックスバッファのフォーマット
<i>m_skinningMatrixFormat</i>	スキニングが有効な場合のスキニング行列のフォーマット (およびサイズ)
<i>m_cacheOptimizerFunc</i>	最終的に分割された三角形リストを RSX®頂点キャッシュ用に最適化するコールバック関数へのポインタ。Edgeは 2 つのオプティマイザ ( <a href="#">edgeGeomKCacheOptimizer()</a> および <a href="#">edgeGeomKCacheOptimizerHillclimber()</a> ) を提供していますが、それ以外の実装 (NvTriStrip など) も簡単にサポートすることができます。NULL の場合、頂点キャッシュの最適化は実行されません。
<i>m_cacheOptimizerUserData</i>	このポインタは、 <i>m_cacheOptimizerFunc</i> の <i>userData</i> 引数として渡されます。データの解釈は、該当するキャッシュオプティマイザに完全に依存します。

<code>m_customDataSizeFunc</code>	Edge 以外のカスタムジョブデータのために、パーティション中に確保される領域のサイズを提供するために使われる、オプションのコールバック関数へのポインタ。コールバックの引数には、カスタムデータバッファサイズを正確に推定できるように、現在構築中のパーティションの要素数の集合が含まれています。この関数は、極めて頻繁に呼び出されるので、計算量が O(N) になるような計算は避けてください。NULL の場合、パーティション中にカスタムデータ用の領域は予約されません。
<code>m_triangleCentroids</code>	<code>m_triangleList</code> に対応する float の配列で、3 つの成分 (x, y, z) によって三角形の重心の位置 (3 つの頂点の平均) を表します。このメンバは、オプション (使わない場合は NULL) ですが、パーティションを空間的により小さくするために、可能ならば指定するようにしてください。
<code>m_skinningMatrixIndexesPerVertex</code>	各頂点の 4 つのスキニングボーンインデックスの配列。使われていないボーンは -1 に設定する必要があります。このメンバは、スキニングされていないジオメトリの場合には、NULL でもかまいません。
<code>m_deltaStreamVertexStride</code>	ブレンド形状ストリーム中の頂点デルタのサイズ (バイト)。ブレンド形状が不要である場合には、0 でもかまいません。
<code>m_blendedVertexIndexes</code>	シーン中の、ゼロ以外のブレンド形状デルタを持つ頂点インデックスすべての配列。NULL でない場合には、パーティショナーは、このデータを利用して、同じパーティション内にブレンドされた頂点とブレンドされていない頂点を混在することを避けます。
<code>m_numBlendedVertexes</code>	<code>m_blendedVertexIndexes</code> 配列中の要素の数

## 解 説

この構造体には、[edgeGeomPartitioner\(\)](#) 関数のパラメータのすべてが含まれています。

## 関 連 項 目

[edgeGeomPartitioner](#)、[EdgeGeomPartitionerOutput](#)、[edgeGeomComputeTriangleCentroids](#)、[edgeGeomGetBlendedVertexes](#)、[edgeGeomKCacheOptimizer](#)、[edgeGeomKCacheOptimizerHillclimber](#)、[EdgeGeomKCacheOptimizerUserData](#)、[EdgeGeomKCacheOptimizerHillclimberUserData](#)

# EdgeGeomPartitionElementCounts

処理中のパーティションの現在の状態を記述する構造体。各パーティション用の Edge 以外のカスタムデータバッファのサイズを計算するために、ユーザコールバックの中で使われます。

## 定 義

```
#include <libedgegeomtool.h>
struct EdgeGeomPartitionElementCounts
{
    uint32_t m_numTriangles;
    uint32_t m_numVertexes;
    uint32_t m_numMatrices;
    uint32_t m_numBlendedVertexes;
};
```

## メ ン バ

<i>m_numTriangles</i>	現在このパーティションに割り当てられている三角形の数
<i>m_numVertexes</i>	現在このパーティションに割り当てられている頂点の数
<i>m_numMatrices</i>	現在このパーティションに割り当てられているスキニング行列の数。この値は、パーティショナーの入力データにおいてスキニングが有効になっていない場合には、ゼロになります。
<i>m_numBlendedVertexes</i>	現在このパーティションに割り当てられている三角形の数。このメンバは、パーティショナーにブレンド形状を渡さない場合には、ゼロになります。

## 解 説

この構造体は、主にパーティショナーによって、現在構築中のパーティションの内容を保持するための内部データとして使われます。この構造体にユーザが会える唯一の場所は、[EdgeGeomCustomPartitionDataSizeFunc\(\)](#) コールバック関数で、このコールバックには、この構造体へのポインタがパラメータとして渡されます。

## 関 連 項 目

[EdgeGeomPartitionerInput](#)

# EdgeGeomPartitionerOutput

[edgeGeomPartitioner](#) () 関数の結果として生成されたパーティション出力を含む構造体

## 定 義

```
#include <libedgegeomtool_partitioner.h>
struct EdgeGeomPartitionerOutput
{
    uint32_t m_numPartitions;
    uint32_t *m_numTrianglesPerPartition;
    uint32_t *m_triangleListOut;
    uint32_t *m_originalVertexIndexesPerPartition;
    uint32_t *m_numUniqueVertexesPerPartition;
    uint32_t *m_ioBufferSizePerPartition;
};
```

## メ ン バ

<code>m_numPartitions</code>	出力構造体中のパーティションの数
<code>m_numTrianglesPerPartition</code>	各パーティションの三角形数の配列。この配列には、 <code>m_numPartitions</code> 個の要素があります。
<code>m_triangleListOut</code>	各三角形の3つの頂点インデックスが1つにパックされた構造の三角形の配列。パーティション0の三角形から始まって、以下パーティション1、2、...の三角形の順に続きます。配列中の要素数の合計は、 <code>m_numTrianglesPerPartition</code> 配列中の値の和です。
<code>m_originalVertexIndexesPerPartition</code>	パーティションのローカルな各頂点を、元の頂点配列のインデックスに対応付ける配列。全パーティションのテーブルは、数珠繋ぎで配置されます。要素数の合計は、 <code>m_numUniqueVertexesPerPartition</code> 配列中の値の和です。
<code>m_numUniqueVertexesPerPartition</code>	各パーティション中の重複しない頂点の数を含む配列。この配列には、 <code>m_numPartitions</code> 個の要素があります。
<code>m_ioBufferSizePerPartition</code>	各パーティションの <code>ioBufferSize</code> の配列。パーティションをDMAを介してSPUに送信する際に、予約するメモリ容量を示します。この配列には、 <code>m_numPartitions</code> 個の要素があります。

## 解 説

これは、[edgeGeomPartitioner](#) () 関数からの出力用の構造体です。この構造体に含まれる配列は、メモリフラグメンテーションを減らすために、全パーティションのデータが隙間なく並べられた、単一のフラットなメモリブロックとして割り当てられます。

この配列のメンバは、呼出し側が、[edgeGeomFree](#) () を使って解放する必要があります。

## 関 連 項 目

[EdgeGeomPartitionerInput](#)

# EdgeGeomSpuVertexAttributeDefinition

SPU への入力として送信される特定の頂点属性のフォーマットを定義するために使われる構造体

## 定 義

```
#include <libedgegeomtool.h>
struct EdgeGeomSpuVertexAttributeDefinition
{
    EdgeGeomInputAttributeType m_type;
    uint32_t m_count;
    EdgeGeomAttributeId m_attributeId;
    uint32_t m_byteOffset;
    uint32_t m_fixedPointBitDepthInteger[4];
    uint32_t m_fixedPointBitDepthFractional[4];
};
```

## メ ン バ

<i>m_type</i>	この属性の変換対象のデータ型
<i>m_count</i>	この属性中の <i>m_type</i> の値の数
<i>m_attributeId</i>	EDGE_GEOM_ATTRIBUTE_ID_* 列挙型の属性を一意に特定する数値
<i>m_byteOffset</i>	入力バッファ中の、伸長ルーチンがこの属性のアンパックを開始するバイトを指定します
<i>m_fixedPointBitDepthInteger</i>	固定小数点変換の整数部分を表すビットの数を示します。この変数が使われるのは、 <i>m_type</i> ==kSpuAttr_FixedPoint の場合だけです。
<i>m_fixedPointBitDepthFractional</i>	固定小数点変換の小数部分を表すビットの数を示します。この変数が使われるのは、 <i>m_type</i> ==kSpuAttr_FixedPoint の場合だけです。

*m\_type* には、以下の値を設定することができます。

列挙型	値	解説
kSpuAttr_I16N	1	[1..1]を1/65535 間隔で正規化した値
kSpuAttr_F32	2	浮動小数点 32 ビット
kSpuAttr_F16	3	浮動小数点 16 ビット: 符号 1 ビット/指数部 5 ビット/仮数部 11 ビット
kSpuAttr_U8N	4	[0..1]を1/255 間隔で正規化した値
kSpuAttr_I16	5	符号付 16 ビット
kSpuAttr_X11Y11Z10N	6	3 成分が正規化された 32 ビット
kSpuAttr_U8	7	符号なし 8 ビット
kSpuAttr_FixedPoint	8	固定小数点フォーマット、ユーザ指定* (SPU のみ。下の解説セクションの制限事項を参照)
kSpuAttr_UnitVector	9	3 成分が正規化されパックされた 24 ビット (SPU のみ)

## 解 説

この構造体は、SPU 用のデータに対応する特定の属性を、ツールがどのように圧縮するかを厳密に定義します。この構造体によって記述できる属性には、たとえば、頂点位置などがあります。任意のフレー

バーは、インタリーブされている属性バッファにパックされた複数の属性を示すために、これらの構造体を1つ以上を含んでいます。

以下は、**注意すべき重要な制限**です。

$(m\_fixedPointBitDepthInteger + m\_fixedPointBitDepthFractional) * m\_count$   
は8で割り切れる必要があります（属性が正しくバイトアラインメントされるようにします）。また、各成分のビット数の合計（整数部+小数部）は、32以下である必要があります。

## **関 連 項 目**

---

[EdgeGeomRsxVertexFormat](#)

# EdgeGeomSpuVertexFormat

SPU に送信される完全な入力ストリームフォーマットを定義する構造体

## 定 義

```
#include <libedgegeomtool.h>
struct EdgeGeomSpuVertexFormat
{
    uint32_t m_numAttributes;
    uint32_t m_vertexStride;
    EdgeGeomSpuVertexAttributeDefinition m_attributeDefinition[16];
};
```

## メ ン バ

<code>m_numAttributes</code>	<code>m_attributeDefinition</code> 配列の中 の <code>EdgeGeomSpuVertexAttributeDefinition</code> 構造体の数。16 以下である 必要があります。
<code>m_vertexStride</code>	このフォーマットに圧縮されたときの 1 頂点当たりのバイト数
<code>m_attributeDefinition</code>	各頂点属性の順序や圧縮メソッドを記述する <a href="#">EdgeGeomSpuVertexAttributeDefinition</a> 構造体の配列

## 解 説

`EdgeGeomSpuVertexFormat` 構造体は、ツール内のジオメトリのコンパイルを管理するために、2 通りの方法で使われます。まず、この構造体は、[edgeGeomPartitioner\(\)](#) 関数により、各パーティションが生成されたときに、SPUが必要とするメモリを測定するために使われます。その後、この構造体は、頂点およびインデックスバッファ生成処理の際に、属性型から圧縮データ形式への変換を制御するために使われます。

[EdgeGeomSpuVertexFormat](#) 構造体は、SPU入力頂点ストリーム、頂点デルタストリーム（ブレンド形状が使われている場合）の両方を記述するために使われます。頂点デルタを記述する際には、さらなる制限が適用されます。Edge頂点デルタは、固定小数点を使ってより少ないビット数に効率的に圧縮できるように、差分形式で保存されます。けれども、（単位ベクトルやX11Y11Z10Nのように）最大範囲[-1..1]をもつデータ形式は、通常意味がありません。反対方向を向く 2 つの単位法線同士を差し引くと、その差分の大きさは、1.0 を超えます。また、頂点デルタフォーマットで `kSpuAttr_UnitVector` 属性型を使用した場合、この大きさは常に 1.0 であるため、ほとんどの場合にエラーとなります。

フォーマットの妥当性を検証する目的で利用できるヘルパー関数 [edgeGeomSpuVertexFormatIsValid\(\)](#) が提供されています。

## 関 連 項 目

[EdgeGeomSpuVertexAttributeDefinition](#)、[edgeGeomSpuVertexFormatIsValid](#)



# EdgeGeomRsxVertexAttributeDefinition

RSX®へ（PPU または SPU から）入力として送信される単一の頂点属性のフォーマットを定義する構造体

## 定 義

```
#include <libedgegeomtool_vertexformat.h>
struct EdgeGeomRsxVertexAttributeDefinition
{
    EdgeGeomOutputAttributeType m_type;
    uint32_t m_count;
    uint32_t m_byteOffset;
    EdgeGeomAttributeId m_attributeId;
};
```

## メ ン バ

*m\_type*                      この属性のデータ型  
*m\_count*                    この属性中の *m\_type* の値の数  
*m\_byteOffset*                それぞれの頂点内において、この属性のバイトを指定します。  
*m\_attributeId*                EDGE\_GEOM\_ATTRIBUTE\_ID\_\* 列挙型の属性を一意に特定する数値。  
*m\_type* には、以下の値を設定することができます。

列挙型	値	解説
kRsxAttr_I16N	1	[1..1]を 1/65535 間隔で正規化した値
kRsxAttr_F32	2	浮動小数点 32 ビット
kRsxAttr_F16	3	浮動小数点 16 ビット： 符号 1 ビット/指数部 5 ビット/仮数部 11 ビット
kRsxAttr_U8N	4	[0..1]を 1/255 間隔で正規化した値
kRsxAttr_I16	5	符号付 16 ビット
kRsxAttr_X11Y11Z10N	6	3 成分が正規化されパックされた 32 ビット
kRsxAttr_U8	7	符号なし 8 ビット

## 解 説

この構造体は、RSX®によってただちに利用できるレンダリング可能なフォーマットの特定の頂点属性のデータを、厳密に定義します。この構造体によって記述できる属性には、たとえば頂点位置などがあります。

RSX®は、SPU のみの単位ベクトルや固定小数点圧縮型をデコードできないので、出力変換では、これらの型は利用できません。出力としてサポートされるのは、RSX®ネイティブの型だけです。

## 関 連 項 目

[EdgeGeomRsxVertexFormat](#)

---

# EdgeGeomRsxVertexFormat

---

RSX®による使用に適した頂点ストリーム形式を定義する構造体

## 定 義

---

```
#include <libedgegeomtool.h>
struct EdgeGeomRsxVertexFormat
{
    unsigned m_numAttributes;
    unsigned m_vertexStride;
    EdgeGeomRsxVertexAttributeDefinition m_attributeDefinition[16];
};
```

## メ ン バ

---

<i>m_numAttributes</i>	<i>m_attributeDefinition</i> 配列中の有効な要素の数。16 以下である必要があります。
<i>m_vertexStride</i>	1 頂点あたりのバイト数
<i>m_attributeDefinition</i>	各頂点属性の順序や圧縮メソッドを記述する <a href="#">EdgeGeomRsxVertexAttributeDefinition</a> 構造体の配列

## 解 説

---

この構造体は、RSX®によって処理される頂点ストリームのフォーマットを記述します。この構造体は、より幅広い頂点属性タイプを扱うことのできる [EdgeGeomSpuVertexFormat](#) のサブセットです。フォーマットの妥当性を検証する目的で利用できるヘルパー関数 [edgeGeomSpuVertexFormatIsValid\(\)](#) が提供されています。

## 関 連 項 目

---

[EdgeGeomRsxVertexAttributeDefinition](#)

# libedgegeomtool関数

# edgeGeomPartitioner

ジオメトリユニットを SPU での利用に適したパーティションに分割します

## 定 義

```
#include <libedgegeomtool_partitioner.h>
void edgeGeomPartitioner(
    const EdgeGeomPartitionerInput_&dataIn,
    EdgeGeomPartitionerOutput_ *dataOut,
)
```

## 呼 出 条 件

マルチスレッドセーフですが、ioBufferCallback の呼出条件に依存します。  
スレッドから呼び出し可能。

## 引 数

<i>dataIn</i>	パーティショナーが必要とするパラメータのすべてが含まれています
<i>dataOut</i>	パーティショナーが出力するデータのすべてが含まれています。出力構造体のフィールドは、呼出し側は、 <a href="#">edgeGeomFree()</a> を使って解放する必要があります

## 返 り 値

この関数には、エラーリターンコードがありません。エラーはすべて EDGEASSERT マクロによってトリガされます。このマクロは、デフォルトで型 (char\*) の例外をスローします。

## 解 説

この関数は、入力ジオメトリデータを、ジオメトリに対して行われる処理や、ジオメトリの圧縮形式が必要とするバイト数を考慮にいれながら、SPU でランタイム処理の際に利用できるメモリ容量に基づいて、1 つまたは複数の出力パーティションに分割します。

このアルゴリズムの核心部分は、以下のように要約できます。

- (1) 割当てられていない三角形の中から「最良」のものを探して、それを現在のパーティションに割り当てます。
- (2) 現在のパーティションの中の三角形と辺を共有する三角形の中で、「最良」のものを探します。
- (3) パーティションのメモリリミットを超えた場合には、最後の三角形を取り除いてから、新しいパーティションを開始し、手順 1 に進みます。それ以外の場合には、手順 2 に進みます。
- (4) 割り当てられていない三角形がなくなったら、部分的に埋まっているパーティションのすべてを保存します。

この収集する三角形のグループを選択するために使われるアルゴリズムは、「貪欲アルゴリズム」の一種です。手順 1 の「最良」は、既存のパーティションの重心に最も距離の近い面として定義されます。

## 関 連 項 目

[EdgeGeomPartitionerInput](#)、[EdgeGeomPartitionerOutput](#)、[edgeGeomFree](#)

---

# edgeGeomAlloc

---

ユーザー定義のコールバックを使ってメモリを割り当てます

## 定 義

---

```
#include <libedgegeomtool.h>
void *edgeGeomAllocEx (
    size_t allocSize,
    const char *filename,
    const uint32_t lineNumber
)
#define edgeGeomAlloc(allocSize) edgeGeomAllocEx(allocSize, __FILE__, __LINE__)
```

## 呼 出 条 件

---

マルチスレッドセーフですが、ユーザコールバックの呼出条件に依存します。  
スレッドから呼び出し可能。

## 引 数

---

`allocSize`      要求された割り当てサイズ (バイト)

## 返 り 値

---

新たに割り当てられたメモリブロックへのポインタ。このポインタは、メモリが不要になったら、[edgeGeomFree\(\)](#)に渡す必要があります。

## 解 説

---

([edgeGeomSetAllocFunc\(\)](#)で指定される) ユーザ定義のメモリ割り当てコールバックのラップです。この関数は、libedgegeomtoolの中で、動的なメモリ割り当てが必要になるたびに、内部的に使われます。  
デフォルトのメモリ割り当てコールバックは `malloc()` です。

## 関 連 項 目

---

[edgeGeomFree](#)、[edgeGeomSetAllocFunc](#)

---

# edgeGeomFree

---

ユーザー定義のコールバックを使ってメモリを解放します

## 定 義

---

```
#include <libedgegeomtool.h>
void edgeGeomFreeEx(
    void *ptr,
    const char *filename,
    const uint32_t lineNumber
)
#define edgeGeomFree(ptr) edgeGeomFreeEx(ptr, __FILE__, __LINE__)
```

## 呼 出 条 件

---

マルチスレッドセーフですが、ユーザコールバックの呼出条件に依存します。  
スレッドから呼び出し可能。

## 引 数

---

*ptr*      解放するメモリブロックへのポインタ。このメモリは、[edgeGeomAlloc\(\)](#)で割り当てられたものである必要があります。

## 返 り 値

---

なし

## 解 説

---

([edgeGeomSetFreeFunc\(\)](#)で指定される) ユーザ定義のメモリ解放コールバックのラップです。この関数は、動的に割り当てられたメモリを解放するたびに、libedgegeomtoolの内部で使われます。また、libedgegeomtoolの関数によって呼び出し側に返された動的割り当てバッファは、すべて [edgeGeomFree\(\)](#) で解放する必要があります。  
デフォルトのメモリ解放コールバックは free() です。

## 関 連 項 目

---

[edgeGeomAlloc](#)、[edgeGeomSetFreeFunc](#)

---

# edgeGeomSetAllocFunc

---

[edgeGeomAlloc\(\)](#) ラッパによって呼び出されるユーザー定義のメモリ割り当てコールバックを指定します

## 定 義

---

```
#include <libedgegeomtool.h>
typedef void* (*EdgeGeomAllocFunc)(size_t allocSize,
    const char *filename, const uint32_t lineNumber);
void edgeGeomSetAllocFunc(
    EdgeGeomAllocFunc func
)
```

## 呼 出 条 件

---

マルチスレッドセーフである。  
スレッドから呼び出し可能。

## 引 数

---

<i>func</i>	必要なメモリ割り当てコールバックへの関数ポインタ。このコールバックは、 <a href="#">edgeGeomFree()</a> によってラップされたメモリの解放コールバックと互換性のあるものである必要があります。
-------------	---

## 返 り 値

---

なし

## 解 説

---

この関数は、libedgegeomtoolの中で内部的に使われるメモリ割り当て関数をオーバーライドするために使われます。カスタムアロケータを使う場合には、かならず、[edgeGeomSetFreeFunc\(\)](#)を使ってメモリ解放コールバックを更新するようにしてください。  
メモリ割り当てコールバックは、デフォルトでは free() に設定されます。

## 関 連 項 目

---

[edgeGeomAlloc](#)

---

# edgeGeomSetFreeFunc

---

[edgeGeomFree\(\)](#) ラッパによって呼び出されるユーザー定義のメモリ解放コールバックを指定します

## 定 義

---

```
#include <libedgegeomtool.h>
typedef void (*EdgeGeomFreeFunc)(void *ptr, const char *filename,
    const uint32_t lineNumber);
void edgeGeomSetFreeFunc(
    EdgeGeomFreeFunc func
)
```

## 呼 出 条 件

---

マルチスレッドセーフである。  
スレッドから呼び出し可能。

## 引 数

---

<i>func</i>	必要なメモリ解放コールバックへの関数ポインタ。このコールバックは、 <a href="#">edgeGeomAlloc()</a> によってラップされたメモリ割り当てコールバックと互換性のあるものである必要があります。
-------------	---

## 返 り 値

---

なし

## 解 説

---

この関数は、libedgegeomtoolの中で内部的に使われるメモリ解放関数をオーバーライドするために使われます。カスタムアロケータを使う場合には、かならず、[edgeGeomSetAllocFunc\(\)](#)を使ってメモリ割り当てコールバックを更新するようにしてください。

メモリ解放コールバックは、デフォルトでは `malloc()` に設定されます。

## 関 連 項 目

---

[edgeGeomFree](#)



---

# edgeGeomGetCommandBufferHoleSize

---

指定された Edge ジオメトリセグメントの描画コマンドが RSX® コマンドバッファ中で必要とする領域の最大値を決定します

## 定 義

---

```
#include <libedgegeomtool.h>
void edgeGeomGetCommandBufferHoleSize (
    uint32_t numOutputAttributes,
    uint32_t numIndexes
)
```

## 呼 出 条 件

---

マルチスレッドセーフである。  
スレッドから呼び出し可能。

## 引 数

---

<i>numOutputAttributes</i>	このセグメントによって RSX® に送信される頂点属性の数
<i>numIndexes</i>	セグメントのインデックスバッファ中の要素の数

## 返 り 値

---

このセグメントをレンダリングするために、Edge ランタイムによって生成される RSX® コマンドデータの最大バイト数。この値は、最も近い 16 の倍数バイトまで切り上げられます。

## 解 説

---

Edge ランタイムは、各セグメントに関連する描画コマンドのために、RSX® コマンドバッファの中に作成するホールの大きさを知る必要があります。コマンドの正確なサイズは、最終的にレンダリングされる頂点属性やカリングされなかった三角形の数によって異なります。この関数は、(三角形がすべてカリングされないと仮定して) 指定されたセグメントが必要とする可能性のある最大領域を計算します。この関数の結果は、[edgeGeomMakeSpuConfigInfo\(\)](#) の引数として渡す必要があります。

## 関 連 項 目

---

[edgeGeomMakeSpuConfigInfo](#)

# edgeGeomGetScratchBufferSizeInQwords

指定されたジオメトリセグメントが SPU 上で必要とするスクラッチ領域を決めます

## 定 義

```
#include <libedgegeomtool.h>
uint32_t edgeGeomGetScratchBufferSizeInQwords (
    uint32_t numInputAttributes,
    uint32_t numUniqueVertexes
)
```

## 呼 出 条 件

マルチスレッドセーフである。  
スレッドから呼び出し可能。

## 引 数

<i>numInputAttributes</i>	すべての SPU 入力ストリーム内の、SPU 入力頂点属性の合計数
<i>numUniqueVertexes</i>	このパーティション・セグメントの中の重複しない頂点の数。この値は、 <a href="#">EdgeGeomPartitionerOutput</a> 構造体から取得することができます。

## 返 り 値

SPU 上で必要になるスクラッチ領域の量を（16 バイトのクワッドワード単位で）返します。

## 解 説

SPU ジョブマネージャ（SPURS など）は、ジョブの入出力バッファとして必要となる領域に加えて、各ジョブが必要とする追加の一時記憶領域の量を知る必要があります。この関数は、指定されたパーティション用のスクラッチ領域を計算します。

必要な領域は、入力頂点属性 1 つごとに一様テーブル 1 つ。カリングが有効な場合、もしくはカスタムブレンド形状フレーバーが使われている場合には、さらに追加テーブル 1 つです。各一様テーブルは、（8 つの倍数個の頂点まで切り上げられた）頂点 1 つごとに 16 バイトのクワッドワード 1 つを含みます。

## 関 連 項 目

[EdgeGeomPartitionerOutput](#)

# edgeGeomGetSpuVertexFormat

指定されたID番号に関連付けられた組み込みのSPU頂点フォーマットへのポインタを取得します

## 定 義

```
#include <libedgegeomtool.h>
EdgeGeomSpuVertexFormat* edgeGeomGetSpuVertexFormat(
    uint32_t formatId
    EdgeGeomSpuVertexFormat *outFormat = NULL
)
```

## 呼 出 条 件

マルチスレッドセーフである（下の例外に注意）。

スレッドから呼び出し可能。

この関数は `edgeGeomAlloc()` を呼び出すが、これは任意のユーザ関数でオーバーライドすることができる。ただし、ユーザの割り当て関数がマルチスレッドセーフでないと、この関数自体もマルチスレッドセーフではなくなる。

## 引 数

<i>formatId</i>	必要な組み込み SPU 頂点フォーマットの ID 番号 ( <code>EDGE_GEOM_SPU_VERTEX_FORMAT_*</code> という名前の列挙型より)
<i>outFormat</i>	NULLでない場合、要求された頂点フォーマットはこのポインタの指す領域にロードされる。NULLである場合、新しい <a href="#">EdgeGeomSpuVertexFormat</a> オブジェクトが割り当てられる。

## 返 り 値

指定されたIDが有効な場合には、指定されたフォーマットオブジェクトへのポインタが返されます。

`outFormat` が NULL の場合、呼出し側は、このオブジェクトを [edgeGeomFree\(\)](#) で削除する必要があります。

ID が無効な場合には、戻り値は NULL になり、`outFormat` の指すオブジェクトは（もしある場合は）変更されません。

## 解 説

この関数を使うのは、SPUによって処理されるストリーム（入力頂点および頂点デルタストリーム）のフォーマットを取得する時だけにしてください。SPU出力ストリームおよびRSX®専用ストリームはフォーマットIDの範囲が異なるので、[dgeGeomGetRsxVertexFormat\(\)](#) を使う必要があります。

## 関 連 項 目

[dgeGeomGetRsxVertexFormat](#)

---

# dgeGeomGetRsxVertexFormat

---

指定された ID 番号に関連付けられた組み込みの RSX®頂点フォーマットへのポインタを取得します

## 定 義

---

```
#include <libedgegeomtool.h>
EdgeGeomRsxVertexFormat* edgeGeomGetRsxVertexFormat(
    uint32_t formatId,
    EdgeGeomRsxVertexFormat *outFormat = NULL
)
```

## 呼 出 条 件

---

マルチスレッドセーフである（下の例外に注意）。

スレッドから呼び出し可能。

この関数は `edgeGeomAlloc()` を呼び出すが、これは任意のユーザ関数でオーバーライドすることができる。ただし、ユーザの割り当て関数がマルチスレッドセーフでないと、この関数自体もマルチスレッドセーフではなくなる。

## 引 数

---

<i>formatId</i>	必要な組み込み RSX®頂点フォーマットの ID 番号 ( <code>EDGE_GEOM_RSX_VERTEX_FORMAT_*</code> という名前の列挙型より)。
<i>outFormat</i>	NULL でない場合、要求された頂点フォーマットはこのポインタの 指す領域にロードされる。NULL である場合、新し い <a href="#">EdgeGeomRsxVertexFormat</a> オブジェクトが割り当てられる。

## 返 り 値

---

指定された ID が有効な場合には、指定されたフォーマットオブジェクトへのポインタが返されます。  
`outFormat` が NULL の場合、呼出し側は、このオブジェクトを [edgeGeomFree\(\)](#) で削除する必要があります。

ID が無効な場合には、戻り値は NULL になり、`outFormat` の指すオブジェクトは（もしある場合は）変更されません。

## 解 説

---

指定された ID 番号に関連付けられた組み込みの RSX®頂点フォーマットへのポインタを取得します。

## 関 連 項 目

---

[edgeGeomGetSpuVertexFormatId](#)

---

# edgeGeomGetSpuVertexFormatId

---

指定された SPU 頂点フォーマットの ID 番号を取得します

## 定 義

---

```
#include <libedgegeomtool.h>
uint8_t edgeGeomGetSpuVertexFormatId(
    const EdgeGeomSpuVertexFormat &format
)
```

## 呼 出 条 件

---

マルチスレッドセーフである。  
スレッドから呼び出し可能。

## 引 数

---

*format* ID を取得する SPU 頂点フォーマット

## 返 り 値

---

指定されたフォーマットが組み込み SPU 頂点フォーマットの 1 つに一致した場合には、その ID 番号 (EDGE\_GEOM\_SPU\_VERTEX\_FORMAT\_\* という列挙型より) が返されます。  
一致するものが見つからない場合には、(カスタムフォーマットを表す) 0xFF という特殊な ID が返されます。

## 解 説

---

この関数を呼び出すのは、SPUによって処理されるストリーム (入力頂点および頂点デルタストリーム) のフォーマットを取得する場合だけにしてください。SPU出力ストリームおよびRSX®専用ストリームはフォーマットIDの範囲が異なるので、[EdgeGeomRsxVertexFormat](#)を使う必要があります。

## 関 連 項 目

---

[edgeGeomGetSpuVertexFormat](#)

---

# edgeGeomGetRsxVertexFormatId

---

指定された RSX®頂点フォーマットの ID 番号を取得します

## 定 義

---

```
#include <libedgegeomtool.h>
uint8_t edgeGeomGetRsxVertexFormatId(
    const REF_Ref166049960 Yh EdgeGeomRsxVertexFormat &format
)
```

## 呼 出 条 件

---

マルチスレッドセーフです。  
スレッドから呼び出すことができます。

## 引 数

---

*format* ID を取得する RSX®頂点フォーマット

## 返 り 値

---

指定されたフォーマットが組み込み RSX®頂点フォーマットの 1 つに一致した場合には、その ID 番号 (EDGE\_GEOM\_RSX\_VERTEX\_FORMAT\_\* 列挙型から選ばれたもの) が返されます。  
一致するものが見つからない場合には、(カスタムフォーマットを表す) 0xFF という特殊な ID が返されます。

## 解 説

---

この関数を利用するのは、RSX®によって処理されるストリームのフォーマットを取得する場合だけにしてください。SPU 出力ストリームの場合、RSX®専用ストリームと範囲の異なるフォーマット ID を持っているので、[edgeGeomGetSpuVertexFormatId](#) を使う必要があります。

## 関 連 項 目

---

[REF\\_Ref166049960 Yh EdgeGeomRsxVertexFormat](#)

---

# edgeGeomSpuVertexFormatIsValid

---

SPU 頂点フォーマットの整合性を確認します

## 定 義

---

```
#include <libedgegeomtool.h>
bool edgeGeomSpuVertexFormatIsValid(
    const EdgeGeomSpuVertexFormat &format
)
```

## 呼 出 条 件

---

スレッドから呼び出し可能。  
マルチスレッドセーフである。

## 引 数

---

*format*      確認を行う SPU 頂点フォーマット

## 返 り 値

---

指定されたフォーマットが、フォーマットの属性について、型、コンポーネント数、バイトオフセットなどのエラーを検出する一連の内部整合性チェックに合格した場合に true を返します。  
テストに失敗した場合には false を返します。

## 解 説

---

この関数は、SPU 頂点フォーマットに対して一連の内部整合性チェックを実行します。  
RSX®頂点フォーマットは [edgeGeomRsxVertexFormatIsValid\(\)](#) を使用しなければなりません。

## 関 連 項 目

---

[EdgeGeomSpuVertexFormat](#)

---

# edgeGeomRsxVertexFormatIsValid

---

RSX®頂点フォーマットの整合性を確認します

## 定 義

---

```
#include <libedgegeomtool.h>
bool edgeGeomRsxVertexFormatIsValid(
    const EdgeGeomRsxVertexFormat &format
)
```

## 呼 出 条 件

---

スレッドから呼び出し可能。  
マルチスレッドセーフである。

## 引 数

---

*format*            確認を行う RSX®頂点フォーマット

## 返 り 値

---

指定されたフォーマットのフォーマット属性について、型、コンポーネント数、バイトオフセットなどのエラーを検出する一連の内部整合性チェックに合格した場合に true を返します。  
テストに失敗した場合には false を返します。

## 解 説

---

この関数は、RSX®頂点フォーマットに対して一連の内部整合性チェックを実行します。

## 関 連 項 目

---

[EdgeGeomSpuVertexFormat](#)



---

# edgeGeomGetSpuVertexAttributeSize

---

頂点ストリーム中の指定された頂点属性の各インスタンスのサイズを（バイトで）返します

## 定 義

---

```
#include <libedgegeomtool.h>
uint32_t edgeGeomGetSpuVertexAttributeSize (
    const EdgeGeomSpuVertexAttributeDefinition &attr
)
```

## 呼 出 条 件

---

スレッドから呼び出し可能。  
マルチスレッドセーフである。

## 引 数

---

*attr*                      サイズを返す SPU 頂点属性

## 返 り 値

---

指定された属性のサイズを（バイトで）返します。

## 解 説

---

この関数は、カスタム頂点フォーマットを構築する際に便利です。

## 関 連 項 目

---

[edgeGeomGetRsxVertexAttributeSize](#)

---

# edgeGeomGetRsxVertexAttributeSize

---

頂点ストリーム中の指定された頂点属性の各インスタンスのサイズを（バイトで）返します

## 定 義

---

```
#include <libedgegeomtool.h>
uint32_t edgeGeomGetRsxVertexAttributeSize (
    const EdgeGeomRsxVertexAttributeDefinition &attr
)
```

## 呼 出 条 件

---

スレッドから呼び出し可能。  
マルチスレッドセーフである。

## 引 数

---

*attr*            サイズを返す RSX 頂点属性

## 返 り 値

---

指定された属性のサイズを（バイトで）返します。

## 解 説

---

この関数は、カスタム頂点フォーマットを構築する際に便利です。

## 関 連 項 目

---

[edgeGeomGetSpuVertexAttributeSize](#)

---

# edgeGeomGetAttributeSlotIndex

---

頂点属性に関連付けられた頂点プログラムスロットインデックスを取得します

## 定 義

---

```
#include <libedgegeomtool.h>
uint8_t edgeGeomGetAttributeSlotIndex(
    EdgeGeomAttributeId attrId
)
```

## 呼 出 条 件

---

スレッドから呼び出し可能。  
マルチスレッドセーフである。

## 引 数

---

*attrId* 頂点プログラムスロットインデックスを取得する対象となる頂点属性の ID

## 返 り 値

---

指定された属性に対する頂点プログラムスロットインデックスを返します。この関数によって返される属性インデックスは、関連する頂点プログラムによって使用されている属性マッピングに一致する必要があります。

## 解 説

---

この関数は [edgeGeomMakeSpuConfigInfo\(\)](#) で内部的に使用されます。最終の正しい頂点プログラムスロットインデックスは、([edgeGeomSetAttributeSlotIndex\(\)](#) を使用して) この関数が呼ばれる前に設定されなければなりません。

## 関 連 項 目

---

[edgeGeomSetAttributeSlotIndex](#)、[edgeGeomMakeSpuConfigInfo](#)

---

# edgeGeomSetAttributeSlotIndex

---

頂点属性に関連付けられた頂点プログラムスロットインデックスを設定します

## 定 義

---

```
#include <libedgegeomtool.h>
void edgeGeomSetAttributeSlotIndex(
    EdgeGeomAttributeId attrId,
    uint8_t slotIndex
)
```

## 呼 出 条 件

---

スレッドから呼び出し可能。  
マルチスレッドセーフである。

## 引 数

---

<i>attrId</i>	頂点プログラムスロットインデックスを設定する頂点属性の ID
<i>slotIndex</i>	指定された属性用の頂点プログラムスロットインデックス。 有効な値は 0-15 の範囲である。

## 返 り 値

---

なし

## 解 説

---

指定された属性に頂点プログラムスロットインデックスを設定します。この関数によって設定された属性インデックスは、関連する頂点プログラムによって使われる属性マッピングと一致している必要があります。

全属性の頂点プログラムスロットインデックスは、[edgeGeomMakeSpuConfigInfo\(\)](#) が呼び出される前に、最終値に設定される必要があります。一般的な頂点属性には、すべて（sce-cgc で使われるマッピングに対応する）適切なデフォルトが提供されるということに注意してください。

Edge の組み込み頂点フォーマット用の頂点プログラムスロットインデックスは、Edge SPU のランタイムコードにハードコードされていることに注意してください。この関数を、Edge の組み込み頂点属性（位置、法線、接線、従法線など）のスロットインデックスの変更に使う場合、ランタイムコード中の対応する値も更新する必要があります（詳しくは `target/src/edge/geom./edgegeom.cpp` の `edgeGeomSetVertexDataArrays()` を参照）。

## 関 連 項 目

---

`edgeGeomGetAttributeSlotIndex`

---

# edgeGeomComputeTriangleCentroids

---

三角形リスト中の各三角形の重心（空間平均）を含む配列を生成します

## 定 義

---

```
#include <libedgegeomtool.h>
void edgeGeomComputeTriangleCentroids(
    const float *vertexes,
    uint32_t numFloatsPerVertex,
    uint16_t positionAttributeIndex,
    const uint32_t *triangles,
    uint32_t numTriangles,
    float **outTriangleCentroids
)
```

## 呼 出 条 件

---

スレッドから呼び出し可能。  
マルチスレッドセーフである。

## 引 数

---

<i>vertexes</i>	シーンのメイン頂点ストリームへのポインタ
<i>numFloatsPerVertex</i>	メイン頂点ストリーム中での各頂点のストライド (32 ビット float)
<i>positionAttributeIndex</i>	各頂点の float ブロック中の位置属性のインデックス
<i>triangles</i>	シーンの三角形リストへのポインタ。この配列は、最低でも <i>numTriangles*3</i> 個の要素を持っている必要があります。
<i>numTriangles</i>	シーンの三角形リスト中の三角形の数
<i>outTriangleCentroids</i>	出力される三角形重心へのポインタは、ここに書き込まれます。この配列には、 <i>numTriangles*3</i> 個の要素が含まれています。このメモリは、呼出し側が、 <a href="#">edgeGeomFree()</a> を使って解放する必要があります。

## 返 り 値

---

なし

## 解 説

---

三角形の重心は、かならずしも必要ではありませんが、Edge パーティショナーが空間的にコヒーレントなパーティションを生成するのに役立ちます。出力配列には、入力リスト中の各三角形の重心が含まれています（重心というのは、三角形の3つの頂点の幾何平均です）。

## 関 連 項 目

---

[EdgeGeomPartitionerInput](#)

# edgeGeomGetBlendedVertexes

最低でも 1 つのブレンド形状に含まれるゼロでないブレンド形状デルタをもつ頂点すべてのインデックスを含む、ソートされた配列を生成します

## 定 義

```
#include <libedgegeomtool.h>
void edgeGeomGetBlendedVertexes (
    const float *vertexDeltas,
    uint32_t numVertexes,
    uint32_t numFloatsPerDelta,
    const EdgeGeomSpuVertexFormat& deltaFormat,
    const uint16_t blendedAttributeIndexes,
    const EdgeGeomAttributeId *blendedAttributeIds,
    uint8_t numBlendedAttributes,
    uint32_t numBlendShapes,
    const uint32_t *triangles,
    uint32_t numTriangles,
    uint32_t **outBlendedVertexIndexes,
    uint32_t *outNumBlendedVertexes
)
```

## 呼 出 条 件

スレッドから呼び出し可能。  
マルチスレッドセーフである。

## 引 数

<i>vertexDeltas</i>	シーン中の頂点デルタの配列へのポインタ。この配列のフォーマットの説明については、 <a href="#">EdgeGeomScene</a> を参照してください。この配列は、最低でも $\text{numVertexes} * \text{numFloatsPerDelta} * \text{numBlendShapes}$ 個の要素を持つ必要があります。
<i>numVertexes</i>	シーン中の頂点の数
<i>numFloatsPerDelta</i>	<i>vertexDeltas</i> 配列中での各デルタのストライド (32 ビット float)
<i>deltaFormat</i>	最終的なデータを生成するために使われる頂点デルタフォーマット。このフォーマット中に存在する属性だけが、ゼロでないデルタデータ検証の対象となります。
<i>blendedAttributeIndexes</i>	<i>vertexDeltas</i> 中の各頂点デルタに対応する float データのブロックの中の、全ブレンド頂点属性インデックスの配列。この配列には、最低でも <i>numBlendedAttributes</i> 個の要素が含まれている必要があります、また要素の順序は、 <i>blendedAttributeIds</i> 配列の要素と同じである必要があります。
<i>blendedAttributeIds</i>	全ブレンド頂点属性の属性 ID の配列。この配列には、最低でも <i>numBlendedAttributes</i> 個の要素が含まれている必要があります、また要素の順序は、 <i>blendedAttributeIndexes</i> 配列の要素と同じである必要があります。
<i>numBlendedAttributes</i>	<i>vertexDeltas</i> 配列中の各頂点デルタ中の属性の数
<i>numBlendShapes</i>	シーン中のブレンド形状ターゲットの数
<i>triangles</i>	シーンの三角形リストへのポインタ。この配列は、最低でも $\text{numTriangles} * 3$ 個の要素を持っている必要があります。
<i>numTriangles</i>	シーン中の三角形の数

<code>outBlendedVertexIndexes</code>	ここには、最低でも 1 つのブレンド形状の中でゼロでないデルタをもつ頂点インデックスの配列が書き込まれます。この配列は、昇順でソートされます。この配列は、最低でも <code>*outNumBlendedVertexes</code> 個の要素を含んでいます。このメモリは、呼出し側が <a href="#">edgeGeomFree()</a> を使って解放する必要があります。
<code>outNumBlendedVertexes</code>	ここには、 <code>*outBlendedVertexIndexes</code> 配列の要素の数が書き込まれます。

## 返 り 値

なし

## 解 説

これは必須ではありませんが、パーティショナーは、ブレンド形状に影響される可能性のある頂点はどれかがわかると、より効率的な結果を生成することができます。このデータを利用すると、パーティショナーは、ほとんどがブレンドされていない頂点であるパーティションにほんの少しだけブレンドされた頂点を混ぜることを避けることができます。パーティションの中にブレンドされた頂点が混ざっていると、パーティション全体をブレンドする必要があるので、極めて非効率的なセグメントになる可能性があります。

この関数は、シーン中の頂点インデックスのうち、最低でも 1 つのブレンド形状においてゼロでないデルタデータを持つもののすべてのソートされたリストを作成します。この配列は、[EdgeGeomPartitionerInput](#) 構造体で使うのに適しています。値がゼロでないかどうかをチェックされるのは、指定された頂点デルタフォーマットの中に存在する頂点属性だけです。

## 関 連 項 目

[EdgeGeomPartitionerInput](#)

# edgeGeomBlendShapeAffectsSegment

指定されたブレンド形状が、指定されたジオメトリセグメント中の任意の頂点に対して、ゼロでないデルタを持つかどうかを判定します

## 定 義

```
#include <libedgegeomtool.h>
bool edgeGeomBlendShapeAffectsSegment(
    const float *shapeDeltas,
    uint32_t numFloatsPerDelta,
    const EdgeGeomSpuVertexFormat& deltaFormat,
    const uint16_t *blendedAttributeIndexes,
    const EdgeGeomAttributeId *blendedAttributeIds,
    uint8_t numBlendedAttributes,
    const uint32_t *originalVertexIndexes,
    uint32_t numUniqueVertexes
)
```

## 呼 出 条 件

スレッドから呼び出し可能。  
マルチスレッドセーフである。

## 引 数

<i>shapeDeltas</i>	テストする形状のブレンド形状デルタへのポインタ。この配列は、最低でも <i>numUniqueVertexes</i> * <i>numFloatsPerDelta</i> 個の要素を持つ必要があります。
<i>numFloatsPerDelta</i>	<i>shapeDeltas</i> 配列中での各デルタのストライド (32 ビット float)
<i>deltaFormat</i>	最終的なデータを生成するために使われる頂点デルタフォーマット。このフォーマット中に存在する属性だけが、ゼロでないデルタデータ検証の対象となります。
<i>blendedAttributeIndexes</i>	<i>shapeDeltas</i> 中の各頂点デルタに対応する float データのブロックの中の、全ブレンド頂点属性インデックスの配列。この配列には、最低でも <i>numBlendedAttributes</i> 個の要素が含まれている必要があります、また要素の順序は、 <i>blendedAttributeIds</i> 配列の要素と同じである必要があります。
<i>blendedAttributeIds</i>	全ブレンド頂点属性の属性 ID の配列。この配列には、最低でも <i>numBlendedAttributes</i> 個の要素が含まれている必要があります、また要素の順序は、 <i>blendedAttributeIndexes</i> 配列の要素と同じである必要があります。
<i>numBlendedAttributes</i> <i>originalVertexIndexes</i>	<i>shapeDeltas</i> 配列中の各頂点デルタ中の属性の数 パーティションローカルな頂点順序から、シーンの元の頂点配列中の頂点インデックスへのマッピングテーブル。このデータは、 <a href="#">EdgeGeomPartitionerOutput</a> 構造体の中で使われます。この配列は、有効な要素を最低でも <i>numUniqueVertexes</i> 個持つ必要があります。
<i>numUniqueVertexes</i>	このパーティション中の重複しない頂点の数



## 返 り 値

---

*originalVertexIndexes* 配列中に、*shapeDeltas* 配列中でゼロでないデルタデータを含む属性を 1 つでも持つ頂点が 1 つでもあれば、true を返します。チェックされるのは、指定された *deltaFormat* の中に存在する属性だけです。

チェックした頂点の中にゼロでないデルタデータが見つからない場合には、false を返します。

## 解 説

---

この関数は、シーンが分割されたあとで、シーンのブレンド形状のうちのどれが各ジオメトリセグメントに関連しているかを判定するために呼び出す必要があります。

# edgeGeomMakeSpuConfigInfo

ジオメトリセグメントの EdgeGeomSpuConfigInfo オブジェクトをメモリバッファ中に構築します。この関数は、シリアルライゼーションや Edge ジオメトリランタイムでの即時処理に適しています

## 定 義

```
#include <libedgegeomtool.h>
void edgeGeomMakeSpuConfigInfo(
    uint32_t numUniqueVertexes,
    uint32_t numTriangles,
    uint32_t numInputAttributes,
    uint8_t segmentFlags,
    EdgeGeomIndexesFlavor indexListType,
    EdgeGeomSkinningFlavor skinType,
    EdgeGeomMatrixFormat skinMatrixFormat,
    uint8_t inputVertexFormatId,
    uint8_t secondaryInputVertexFormatId,
    uint8_t outputVertexFormatId,
    uint8_t vertexDeltaFormatId,
    uint32_t commandBufferHoleSize,
    uint8_t **outSpuConfigInfo
)
```

## 呼 出 条 件

スレッドから呼び出し可能。  
マルチスレッドセーフである。

## 引 数

<i>numUniqueVertexes</i>	このジオメトリセグメント中の重複しない頂点の数
<i>numTriangles</i>	このセグメントの三角形リスト中の三角形の数
<i>numInputAttributes</i>	SPU に送信される頂点属性の数。二次頂点ストリームが存在する場合には、その中の属性も含まれます。
<i>segmentFlags</i>	予約（ゼロである必要があります）
<i>indexListType</i>	このジオメトリセグメントで使われているインデックスリストのフォーマット。このセグメントを構築するために使われた値と一致している必要があります。
<i>skinType</i>	このセグメントで使われているスキニングアルゴリズム。このセグメントを構築するために使われた値と一致している必要があります。
<i>skinMatrixFormat</i>	スキニングが有効な場合のスキニング行列のフォーマット（およびサイズ）
<i>inputVertexFormatId</i>	（組み込み SPU 頂点フォーマットのリストからの）このセグメントの一次入力頂点ストリームフォーマットの数値 ID。ID 番号 255（0xFF）は、カスタム入力フォーマットを表します。

---

<code>secondaryInputVertexFormatId</code>	(組み込み RSX®頂点フォーマットのリストからの) このセグメントの二次入力頂点ストリームフォーマットの数値 ID。ID 番号 255 (0xFF) は、カスタム入力フォーマットを表します。このフィールドは、入力ストリームが 1 つしか存在しない場合には無視されます (値は任意でよい)。
<code>outputVertexFormatId</code>	(組み込み RSX®頂点フォーマットのリストからの) このセグメントの出力頂点ストリームフォーマットの数値 ID。ID 番号 255 (0xFF) は、カスタム出力フォーマットを表します。
<code>vertexDeltaFormatId</code>	(組み込み SPU 頂点フォーマットのリストからの) このセグメントのブレンド形状頂点デルタストリームフォーマットの ID 番号。ID 番号 255 (0xFF) は、カスタムフォーマットを表します。このフィールドは、ブレンド形状が存在しない場合には無視されます (値は任意でよい)。
<code>commandBufferHoleSize</code> <code>outSpuConfigInfo</code>	コマンドバッファのサイズ (バイト) ここには、最終的なEdgeGeomSpuConfigInfoオブジェクトへのポインタが書き込まれます。このバッファは、ビッグエンディアン形式にバイトスワップ済みです。このメモリは、呼出し側が、 <a href="#">edgeGeomFree ()</a> を使って解放する必要があります。

## 返 り 値

---

なし

## 解 説

---

セグメントの入出力/デルタフォーマットのID番号を取得するには [edgeGeomGetSpuVertexFormatId \(\)](#) と [edgeGeomGetRsxVertexFormatId \(\)](#) の関数を使います。  
コマンドバッファホールのサイズを求めるには、[edgeGeomGetCommandBufferHoleSize \(\)](#) 関数を使います。

## 関 連 項 目

---

[edgeGeomGetSpuVertexFormatId](#)、[edgeGeomGetCommandBufferHoleSize](#)

---

# edgeGeomMakeIndexBuffer

---

ジオメトリセグメントの最終的なインデックスバッファを構築します。この関数は、シリアルイゼーションや Edge ジオメトリランタイムでの即時処理に適しています

## 定 義

---

```
#include <libedgegeomtool.h>
void edgeGeomMakeIndexBuffer(
    const uint32_t *triangles,
    uint32_t numTriangles,
    EdgeGeomIndexesFlavor indexListType,
    uint8_t **outIndexBuffer,
    uint16_t outIndexDmaSizes[2]
)
```

## 呼 出 条 件

---

スレッドから呼び出し可能。  
マルチスレッドセーフである。

## 引 数

---

<i>triangles</i>	このジオメトリセグメントの三角形リスト。最低でも <i>numTriangles</i> *3 個の要素を含んでいる必要があります。
<i>numTriangles</i>	このセグメントの三角形リスト中の三角形の数
<i>indexListType</i>	このジオメトリセグメンで使われているインデックスリストのフォーマット。このセグメントを構築するために使われた値と一致している必要があります。
<i>outIndexBuffer</i>	ここには、最終的なインデックスバッファへのポインタが書き込まれます。このバッファは、ビッグエンディアン形式にバイトスワップ済みです。このメモリは、呼出し側が、 <a href="#">edgeGeomFree()</a> を使って解放する必要があります。
<i>outIndexDmaSizes</i>	ここには、インデックスバッファの DMA タグのサイズが保存されます。

## 返 り 値

---

なし

## 解 説

---

この関数は、Edge パーティショナーによって生成された各セグメントの三角形リストから、最終的な圧縮インデックスバッファを生成するために使われます。

## 関 連 項 目

---

[EdgeGeomPartitionerOutput](#)

# edgeGeomMakeSpuVertexBuffer

頂点バッファを構築します。この関数は、Edge ジオメトリ SPU ランタイムでの処理に適しています

## 定 義

```
#include <libedgegeomtool.h>
void edgeGeomMakeSpuVertexBuffer (
    const float *sourceVertexes,
    uint32_t numFloatsPerSourceVertex,
    const uint16_t *sourceAttributeIndexes,
    const EdgeGeomAttributeId *sourceAttributeIds,
    uint8_t numSourceAttributes,
    const uint32_t *originalVertexIndexes,
    uint32_t numUniqueVertexes,
    const EdgeGeomSpuVertexFormat &vertexFormat,
    uint8_t **outVertexBuffer,
    uint16_t outVertexDmaSizes[3],
    uint32_t **outFixedPointOffsets,
    uint32_t *outFixedPointOffsetsSize
)
```

## 呼 出 条 件

スレッドから呼び出し可能。  
マルチスレッドセーフである。

## 引 数

<i>sourceVertexes</i>	シーンの元の頂点ストリームへのポインタ
<i>numFloatsPerSourceVertex</i>	<i>sourceVertexes</i> 配列中での各デルタのストライド (32 ビット float)
<i>sourceAttributeIndexes</i>	<i>sourceVertexes</i> 中の各頂点に対応する float データのブロックの中の、全頂点属性インデックスの配列。この配列には、最低でも <i>numSourceAttributes</i> 個の要素が含まれている必要があり、また要素の順序は、 <i>sourceAttributeIds</i> 配列の要素と同じでなくてはなりません。
<i>sourceAttributeIds</i>	全頂点属性の属性 ID の配列。この配列には、最低でも <i>numSourceAttributes</i> 個の要素が含まれている必要があり、また要素の順序は、 <i>sourceAttributeIndexes</i> 配列の要素と同じでなくてはなりません。
<i>numSourceAttributes</i>	<i>sourceVertexes</i> 配列中の各頂点中の属性の数。
<i>originalVertexIndexes</i>	パーティションローカルな頂点順序から、シーンの元の頂点配列中の頂点インデックスへのマッピングテーブル。このデータは、 <a href="#">EdgeGeomPartitionerOutput</a> 構造体の中で使われます。この配列は、有効な要素を最低でも <i>numUniqueVertexes</i> 個持つ必要があります。
<i>numUniqueVertexes</i>	このパーティション中の重複しない頂点の数。
<i>vertexFormat</i>	このセグメントの頂点バッファを生成するために使われる頂点フォーマット

---

<code>outVertexBuffer</code>	ここには、最終的な頂点バッファへのポインタが書き込まれます。このバッファは、ビッグエンディアン形式にバイトスワップ済みです。このメモリは、呼出し側が、 <a href="#">edgeGeomFree()</a> を使って解放する必要があります。
<code>outVertexDmaSizes</code>	ここには、インデックスバッファの DMA タグのサイズが格納されます。
<code>outFixedPointOffsets</code>	ここには、固定小数点オフセットのテーブルが書き込まれます。このテーブルは、 <code>inputFlavor</code> 中の固定小数点属性 1 つごとに 4 つの要素を持ちます。このメモリは、呼出し側が、 <a href="#">edgeGeomFree()</a> を使って解放する必要があります。
<code>outFixedPointOffsetsSize</code>	ここには、 <code>outFixedPointOffsets</code> 配列の要素の数が書き込まれます。 <code>inputFlavor</code> 中の固定小数点頂点属性 1 つごとに 4 個出力されます。

## 返 り 値

---

なし

## 解 説

---

この関数は、Edge パーティショナーを実行して各セグメント中に存在する重複しない頂点を求めた後で、ジオメトリセグメントの最終的な圧縮 SPU 入力頂点バッファを生成するために使われます。

この関数は、一次 SPU 入力頂点ストリームと二次 SPU 入力頂点ストリームのどちらに対しても利用できます。違うのは `vertexFormat` の値だけです。

## 関 連 項 目

---

[EdgeGeomPartitionerOutput](#)

# edgeGeomMakeRsxVertexBuffer

RSX®によるレンダリングに適した頂点バッファを構築します

## 定 義

```
#include <libedgegeomtool.h>
void edgeGeomMakeRsxVertexBuffer(
    const float *sourceVertexes,
    uint32_t numFloatsPerSourceVertex,
    const uint16_t *sourceAttributeIndexes,
    const EdgeGeomAttributeId *sourceAttributeIds,
    uint8_t numSourceAttributes,
    const uint32_t *originalVertexIndexes,
    uint32_t numUniqueVertexes,
    const EdgeGeomRsxVertexFormat &vertexFormat,
    bool writeBigEndian,
    uint8_t **outVertexBuffer,
    uint16_t outVertexDmaSizes[3]
)
```

## 呼 出 条 件

スレッドから呼び出し可能。  
マルチスレッドセーフである。

## 引 数

<i>sourceVertexes</i>	シーンのオリジナルの頂点ストリームへのポインタ
<i>numFloatsPerSourceVertex</i>	<i>sourceVertexes</i> 配列の各差分のストライド (32 ビット float)
<i>sourceAttributeIndexes</i>	<i>sourceVertexes</i> に含まれる、各頂点の float データブロックに含まれる、すべての頂点属性のインデックス配列。この配列には少なくとも <i>numSourceAttributes</i> 個のエントリが含まれなければならない、この配列は <i>sourceAttributeIds</i> 配列のエントリと同じ順序で並べられなければなりません。
<i>sourceAttributeIds</i>	すべての頂点属性に対する属性 ID の配列。この配列には少なくとも <i>numSourceAttributes</i> 個のエントリが含まれなければならない、この配列は <i>sourceAttributeIndexes</i> 配列のエントリと同じ順序で並べられなければなりません。
<i>numSourceAttributes</i>	<i>sourceVertexes</i> 配列に含まれる、各頂点の属性の数。
<i>originalVertexIndexes</i>	シーンのオリジナルの頂点配列の頂点インデックスに対する、パーティションローカルの頂点順序のマッピングテーブル。このデータは <a href="#">EdgeGeomPartitionerOutput</a> 構造体に含まれている場合があります。この配列には少なくとも <i>numUniqueVertexes</i> 個の有効なエントリがなければなりません。
<i>numUniqueVertexes</i>	パーティションに含まれる一意の頂点の数。
<i>vertexFormat</i>	このセグメントの頂点バッファを生成するために使用するべき頂点フォーマット
<i>writeBigEndian</i>	true の場合は、出力バッファはビッグエンディアン (PlayStation®3 ネイティブ) 形式で記録されます。false の場合、ストリームはリトルエンディアン (x86 ネイティブ) 形式で記録されます。

---

<code>outVertexBuffer</code>	最後の頂点バッファへのポインタが書き込まれます。バッファはすでにビッグエンディアン形式にバイト交換されています。呼び出し元は <a href="#">edgeGeomFree()</a> を用いてこのメモリを解放する責任を負います。
<code>outVertexDmaSizes</code>	頂点バッファの DMA タグのサイズがここに格納されます。

## 返 り 値

---

なし

## 解 説

---

この関数は、Edge パーティショナーを実行して各セグメント中に存在する重複しない頂点を求めた後で、ジオメトリセグメントの最終的な圧縮 RSX®入力頂点バッファを生成するために使われます。これらの属性は実行時に Edge に渡されることはありませんが、このセグメントの SPU 入力ストリームと同じ順序で格納されます。

また、この関数は Edge とは関係のない、通常の RSX®専用ストリームを生成する目的にも使用できます。この場合、`numUniqueVertexes` の値は入力頂点の総数と等しくなければならず、`originalVertexIndexes` は `[0, 1, 2, ..., numUniqueVertexes-1]` の恒等写像である必要があります。

## 関 連 項 目

---

[EdgeGeomPartitionerOutput](#)



# edgeGeomMakeSkinningBuffer

ジオメトリセグメントの最終的なスキニングバッファを構築します。この関数は、シリアルライゼーションや Edge ジオメトリランタイムでの即時処理に適しています

## 定 義

```
#include <libedgegeomtool.h>
void edgeGeomMakeSkinningBuffer(
    const int32_t *matrixIndexes,
    const float *skinningWeights,
    EdgeGeomSkinningFlavor skinType,
    EdgeGeomMatrixFormat skinMatrixFormat,
    const uint32_t *originalVertexIndexes,
    uint32_t numUniqueVertexes,
    uint8_t **outSkinIndexesAndWeights,
    uint16_t outSkinIndexesAndWeightsDmaSizes[2],
    uint16_t outSkinMatricesByteOffsets[2],
    uint16_t outSkinMatricesSizes[2]
)
```

## 呼 出 条 件

スレッドから呼び出し可能。  
マルチスレッドセーフである。

## 引 数

<i>matrixIndexes</i>	シーンのボーン行列インデックスの配列へのポインタ。この配列には、頂点 1 つにつき 4 つの要素が含まれており、使われていないインフルエンスは-1 に設定されています。
<i>skinningWeights</i>	シーンのスキニング重みの配列へのポインタ。この配列には、範囲 0.0~1.0 の要素が頂点 1 つにつき 4 つずつ含まれ、各頂点の 4 つの重みの合計は 1.0 になります。
<i>skinType</i>	このセグメントで使われるスキニングアルゴリズム。このアルゴリズムは、先にパーティショナーに渡されて使われたものと一致する必要があります。
<i>skinMatrixFormat</i> <i>originalVertexIndexes</i>	スキニング行列のフォーマット（およびサイズ）。パーティションローカルな頂点順序から、シーンの元の頂点配列中の頂点インデックスへのマッピングテーブル。このデータは、 <a href="#">EdgeGeomPartitionerOutput</a> 構造体の中で使われます。この配列は、有効な要素を最低でも <i>numUniqueVertexes</i> 個持つ必要があります。
<i>numUniqueVertexes</i> <i>outSkinIndexesAndWeights</i>	このパーティション中の重複しない頂点の数。ここには、最終的なスキニングインデックス・重みバッファへのポインタが書き込まれます。このバッファは、ビッグエンディアン形式にバイトスワップ済みです。このメモリは、呼出し側が、 <a href="#">edgeGeomFree()</a> を使って解放する必要があります。

---

<code>outSkinIndexesAndWeightsDmaSizes</code>	ここには、スキニング重み・インデックスバッファ DMA タグのサイズが格納されます。
<code>outSkinMatricesByteOffsets</code>	ここには、このセグメントによって使われる 2 つのスキニング行列範囲の先頭へのバイトオフセットが保存されます。
<code>outSkinMatricesSizes</code>	ここには、このセグメントによって使われる 2 つの行列範囲のサイズが格納されます (バイト)。

## 返 り 値

---

なし

## 解 説

---

この関数は、Edge パーティショナーを実行して各セグメント中に存在する重複しない頂点を求めた後で、ジオメトリセグメントの最終的な圧縮スキニングバッファを生成するために使われます。

Edge ツールは、シーンの全スキニング行列をジオメトリセグメントといっしょに SPU に送信するよりも、各セグメントが使っている行列のすべてを含むボーン配列の 2 つの部分範囲を探します。その部分範囲のオフセットおよびサイズを返すのがこの関数です。この 2 つの範囲は、実行時に PPU によって (ここで示した順序で) 順次 SPU に転送され、連続する単一の行列バッファが作成されます。この関数によって生成された (スキニング重み・インデックス配列中の) 行列インデックスは、この最終的な連続テーブルに関連付けられています。

## 関 連 項 目

---

[EdgeGeomPartitionerOutput](#)

# edgeGeomMakeBlendShapeBuffer

ジオメトリセグメントの特定のブレンド形状のための最終的なブレンド形状バッファを構築します。この関数は、シリアライゼーションや Edge ジオメトリランタイムでの即時処理に適しています

## 定 義

```
#include <libedgegeomtool.h>
void edgeGeomMakeBlendShapeBuffer(
    const float *shapeDeltas,
    uint32_t numFloatsPerDelta,
    const EdgeGeomSpuVertexFormat& deltaFormat,
    const uint16_t *blendedAttributeIndexes,
    const EdgeGeomAttributeId *blendedAttributeIds,
    uint8_t numBlendedAttributes,
    const uint32_t *originalVertexIndexes,
    uint32_t numUniqueVertexes,
    uint8_t **outShapeBuffer,
    uint16_t *outShapeBufferSize
)
```

## 呼 出 条 件

スレッドから呼び出し可能。  
マルチスレッドセーフである。

## 引 数

<i>shapeDeltas</i>	テストする形状のブレンド形状デルタへのポインタ。この配列は、最低でも <i>numUniqueVertexes</i> * <i>numFloatsPerDelta</i> 個の要素を持つ必要があります。
<i>numFloatsPerDelta</i>	<i>shapeDeltas</i> 配列中での各デルタのストライド (32 ビット float)
<i>deltaFormat</i>	最終的なデータを生成するために使われるブレンド形状頂点デルタフォーマット。このフォーマット中に存在する属性だけが、ゼロでないデルタデータ検証の対象となります。
<i>blendedAttributeIndexes</i>	<i>shapeDeltas</i> 中の各頂点デルタに対応する float データのブロックの中の、全ブレンド頂点属性インデックスの配列。この配列には、最低でも <i>numBlendedAttributes</i> 個の要素が含まれている必要があり、また要素の順序は、 <i>blendedAttributeIds</i> 配列の要素と同じでなければなりません。
<i>blendedAttributeIds</i>	全ブレンド頂点属性の属性 ID の配列。この配列には、最低でも <i>numBlendedAttributes</i> 個の要素が含まれている必要があり、また要素の順序は、 <i>blendedAttributeIndexes</i> 配列の要素と同じでなければなりません。
<i>numBlendedAttributes</i>	<i>shapeDeltas</i> 配列中の各頂点デルタ中の属性の数
<i>originalVertexIndexes</i>	パーティションローカルな頂点順序から、シーンの元の頂点配列中の頂点インデックスへのマッピングテーブル。このデータは、 <a href="#">EdgeGeomPartitionerOutput</a> 構造体の中で使われます。この配列は、有効な要素を最低でも <i>numUniqueVertexes</i> 個持つ必要があります。

---

<code>numUniqueVertexes</code>	このパーティション中の重複しない頂点の数
<code>outShapeBuffer</code>	ここには、最終的なブレンド形状バッファへのポインタが書き込まれます。このバッファは、ビッグエンディアン形式にバイトスワップ済みです。このメモリは、呼出し側が、 <a href="#">edgeGeomFree()</a> を使って解放する必要があります。
<code>outShapeBufferSize</code>	ここには、ブレンド形状バッファのサイズが格納されます。

## 返 り 値

---

なし

## 解 説

---

この関数は、Edge パーティショナーを実行して各セグメント中に存在する重複しない頂点を求めた後で、ジオメトリセグメントの最終的な圧縮ブレンド形状バッファを生成するために使われます。シーン中の各セグメントに適用される全ブレンド形状に対してブレンド形状バッファを作成するよりも、[edgeGeomBlendShapeAffectsSegment\(\)](#) 関数を使って、各セグメントに実際に影響を与えるブレンド形状だけを含むようにすることをお勧めします。

## 関 連 項 目

---

[EdgeGeomPartitionerOutput](#)、[edgeGeomBlendShapeAffectsSegment](#)

---

## edgeGeomMergeIdenticalVertexes

---

特定の [EdgeGeomScene](#) オブジェクトから重複した頂点をすべて特定して削除します。変更されるのは三角形リストだけです。実際の頂点データは、元の状態のまま維持されます

### 定 義

---

```
#include <libedgegeomtool_wrap.h>
uint32_t edgeGeomMergeIdenticalVertexes (
    EdgeGeomScene &edgeScene
)
```

### 呼 出 条 件

---

スレッドから呼び出し可能。  
マルチスレッドセーフである。

### 引 数

---

<code>edgeScene</code>	処理に入力されるシーン。このシーンの三角形リストは、等しい頂点への参照がすべて同じ頂点を参照するように変更されます。
------------------------	--

### 返 り 値

---

重複を理由に削除された頂点の数を返します。

### 解 説

---

この関数では、重複する頂点を、クローズドハッシュテーブルを使って求めます。ハッシュキーは、頂点に関連付けられた浮動小数点属性の単純な合計です。この関数は、特定の入力フレーバーを考慮に入れたり、微妙に異なる2つの頂点が圧縮・変換の後に一致するかどうかを検出したりしないので、理想的な方法とは言えません。この関数は、単にあまり良く書かれていないエクスポートやアセットパイプライン変換ルーチンが、必要以上にデータを重複されることを防ぐだけです。  
この関数はパーティションの前に呼び出しておかなければなりません。

# edgeGeomPartitionSceneIntoSegments

入力シーンをジオメトリセグメントのリストに変換するために必要なエンドツーエンドの処理のすべてを実行します。この処理には、バッチへの分割、バッチのセグメントへの分割、各セグメントに必要なデータバッファの生成などが含まれます

## 定 義

```
#include <libedgegeomtool_wrap.h>
void edgeGeomPartitionSceneIntoSegments(
    const EdgeGeomScene &edgeScene,
    const EdgeGeomSegmentFormat &edgeFormat,
    EdgeGeomSegment **outSegments,
    uint32_t *outNumSegments
    EdgeGeomCustomPartitionDataSizeFunc partitionDataSizeFunc = 0,
    EdgeGeomCustomCommandBufferHoleSizeFunc commandBufferHoleSizeFunc = 0
)
```

## 呼 出 条 件

スレッドから呼び出し可能。  
マルチスレッドセーフである。

## 引 数

<i>edgeScene</i>	処理に入力されるシーン
<i>edgeFormat</i>	出力セグメントのフォーマット情報が含まれます。
<i>outSegments</i>	ここには、出力ジオメトリセグメントの配列が書き込まれます。この配列には、 <i>*outNumSegments</i> 個の要素があります。この配列（および各セグメントの全配列メンバ）は、呼出し側が、 <a href="#">edgeGeomFree()</a> を使って解放する必要があります。
<i>outNumSegments</i>	ここには、 <i>outSegments</i> 配列の要素の数が書き込まれます。
<i>partitionDataSizeFunc</i>	パーティションに加えたい（デフォルトの Edge データ以外の）任意の追加データを登録するためのコールバック（オプション）。不要な場合には NULL でもかまいません。
<i>commandBufferHoleSizeFunc</i>	Edge ジョブから呼び出される（Edge の既定のもの以外の）追加の GCM コマンドを登録するためのコールバック（オプション）。不要な場合には NULL でもかまいません。

## 返 り 値

なし。

## 解 説

この関数は、Edge API 関数のラップで、Edge ジオメトリの処理に対して、簡単かつオールインワンのインタフェースを提供します。多くのアプリケーションでは、呼び出す必要のある Edge ジオメトリ関数が、このツールだけの場合もあります。

この関数はまず、シーンをバッチ（シーンのうち特定のマテリアルIDとスキニング設定を使っている部分）に分割します。各バッチは、[edgeGeomPartitioner\(\)](#)に渡されて、ジオメトリセグメントに分割されます。最後に、edgeGeomMake\*関数を使って、各ジオメトリセグメント用のデータバッファが生成されます。

呼出し元は、outSegments配列自体を解放する前に、[edgeGeomFreeSegmentData\(\)](#)を使って、各 [EdgeGeomSegment](#) のデータバッファのメモリを解放するように注意する必要があります。

Edge ジョブが、追加データや、コマンドバッファホールに追加の GCM コマンドを書き込むことを必要とする実行時カスタム処理を行う際には、このような追加の要件をパーティショナーに通知するために、オプションの2つのコールバック関数（*partitionDataSizeFunc*、*commandBufferHoleSizeFunc*）を使います。標準の Edge データテーブルや、最終的な三角形を描画するために必要であるか、Edge のランタイム関数によって内部的に呼び出されるコマンドバッファホールの GCM コマンドでは、このような関数を使う必要はありません。

## **関 連 項 目**

---

[edgeGeomMergeIdenticalVertexes](#)、[edgeGeomFreeSegmentData](#)

---

# edgeGeomFreeSegmentData

---

[EdgeGeomSegment](#)オブジェクトに関連する全データのメモリを解放します（ただし、オブジェクト自体は除く）

## 定 義

---

```
#include <libedgegeomtool_wrap.h>
void edgeGeomFreeSegmentData (
    EdgeGeomSegment &segment
)
```

## 呼 出 条 件

---

スレッドから呼び出し不可。  
マルチスレッドセーフでない。

## 引 数

---

<i>segment</i>	データを削除するセグメント
----------------	---------------

## 返 り 値

---

なし。

## 解 説

---

この関数は、[EdgeGeomSegment](#)構造体中の各配列に対して [edgeGeomFree\(\)](#) を呼び出すだけです。将来、[EdgeGeomSegment](#)構造体のコードに対して変更があってもいいように、ユーザ自身がセグメントのバッファを解放する代わりに、この関数を呼び出すことを推奨します。

## 関 連 項 目

---

[edgeGeomFree](#)



# edgeGeomTriangulatePolygons

任意のポリゴンリストを三角形に変換します

## 定 義

```
#include <libedgegeomtool_wrap.h>
void edgeGeomTriangulatePolygons(
    const int32_t *polygons,
    const int32_t *materialIdPerPolygon,
    uint32_t **outTriangles,
    int32_t **outMaterialIdPerTriangle,
    uint32_t *outNumTriangles
)
```

## 呼 出 条 件

スレッドから呼び出し可能。  
マルチスレッドセーフである。

## 引 数

<i>polygons</i>	頂点インデックスの配列です。各ポリゴンの最後には-1、リストの最後にはさらに-1 が挿入されます。
<i>materialIdPerPolygon</i>	マテリアル ID の配列。ポリゴンごとに ID が 1 つずつ指定されます。この ID は、マテリアルの一意な識別子（ファイル名のハッシュ、ポイントなど）であればなんでもかまいません。
<i>outTriangles</i>	ここには、出力される三角形リストへのポインタが書き込まれます。この配列には、 <i>*outNumTriangles * 3</i> 個の要素があります。このメモリは、呼出し側が <a href="#">edgeGeomFree()</a> を使って解放する必要があります。
<i>outMaterialIdPerTriangle</i>	出力される各三角形のマテリアルIDへのポインタ。この配列には、 <i>*outNumTriangles</i> 個の要素があります。このメモリは、呼出し側が <a href="#">edgeGeomFree()</a> を使って解放する必要があります。
<i>outNumTriangles</i>	ここには、出力配列中の三角形の数が書き込まれます。

## 返 り 値

なし。

## 解 説

この関数は、ポリゴンの最初の頂点からファンニングすることによって、ポリゴンを三角形に分割します。ファンニングが複雑なポリゴン进行处理するための理想的な方法ではないことは広く知られていますが、便宜的にこの関数が用意されています。凸でなおかつ正多角形に近いポリゴンが多いようなジオメトリでは、この方法で十分です。

正しくフォーマットされた入力データストリームは、以下のようになります。

*polygons*

0	1	2	3	-1	1	0	4	5	-1	-1
---	---	---	---	----	---	---	---	---	----	----

*materialIdPerPolygon*

0	0
---	---

# edgeGeomMakeSpuStreamDescription

EdgeGeomSpuVertexFormatのストリーム記述構造体を生成します

## 定 義

```
#include <libedgegeomtool.h>
void edgeGeomMakeSpuStreamDescription (
    const EdgeGeomSpuVertexFormat &vertexFormat,
    uint8_t **outStreamDescription,
    uint16_t *outStreamDescriptionSize
)
```

## 呼 出 条 件

スレッドから呼び出し可能。  
マルチスレッドセーフである。

## 引 数

<i>vertexFormat</i>	ストリーム記述の構築元になる SPU 頂点フォーマット
<i>outStreamDescription</i>	ここには、新しいストリーム記述へのポインタが書き込まれます。 このメモリは、呼出し側が <a href="#">edgeGeomFree()</a> を使って解放する必要があります。
<i>outStreamDescriptionSize</i>	ここには、新しいストリーム記述構造体のサイズ（バイト）が書き込まれます。

## 返 り 値

なし。

## 解 説

この関数は、指定された SPU 頂点フォーマットのストリーム記述バッファを生成します。

## 注 意

この関数によって生成されるバッファは、Edge ジオメトリランタイムによって、入力頂点ストリームを、ジェネリックなデータ駆動アルゴリズムを使って伸張するために使われます。この方法は、さまざまな頂点フォーマットを提供することにより、ユーザーに柔軟性を提供します。それでも、頂点ストリームフォーマットを少ししか使わないユーザーには、コンパイル済みフレーバーを使うことをお勧めします。Edge 頂点圧縮マクロによって生成されるコードは、ストリーム記述を使うデータ駆動圧縮関数よりはるかに効率的です。

---

# edgeGeomMakeRsxStreamDescription

---

EdgeGeomRsxVertexFormatのストリーム記述構造体を生成します

## 定 義

---

```
#include <libedgegeomtool.h>
void edgeGeomMakeRsxStreamDescription(
    const EdgeGeomRsxVertexFormat &vertexFormat,
    uint8_t **outStreamDescription,
    uint16_t *outStreamDescriptionSize
)
```

## 呼 出 条 件

---

スレッドから呼び出し可能。  
マルチスレッドセーフである。

## 引 数

---

<i>vertexFormat</i>	ストリーム記述の構築元になる RSX®頂点フォーマット
<i>outStreamDescription</i>	ここには、新しいストリーム記述へのポインタが書き込まれます。 このメモリは、呼出し側が <a href="#">edgeGeomFree()</a> を使って解放する必要があります。
<i>outStreamDescriptionSize</i>	ここには、新しいストリーム記述構造体のサイズ（バイト）が書き込まれます。

## 返 り 値

---

なし。

## 解 説

---

この関数は、指定された RSX®頂点フォーマットのストリーム記述バッファを生成します。

## 注 意

---

この関数によって生成されるバッファは、Edge ジオメトリランタイムによって、出力頂点ストリームを、ジェネリックなデータ駆動アルゴリズムを使って圧縮するために使われます。この方法は、さまざまな頂点フォーマットを提供することにより、ユーザーに柔軟性を提供します。それでも、頂点ストリームフォーマットを少ししか使わないユーザーには、コンパイル済みフレーバーを使うことをお勧めします。Edge 頂点圧縮マクロによって生成されるコードは、ストリーム記述を使うデータ駆動圧縮関数よりはるかに効率的です。

# edgeGeomKCacheOptimizer

三角形および三角形中の頂点の順序を並べ替え、GPU 上でのキャッシュミス进行最小化します

## 定 義

```
#include <libedgegeomtool.h>
void edgeGeomKCacheOptimizer(
    const uint32_t *inTriangles,
    uint32_t numTriangles,
    void *userData,
    uint32_t *outTriangles
)
```

## 呼 出 条 件

スレッドから呼び出し可能。  
マルチスレッドセーフである。

## 引 数

<i>inTriangles</i>	三角形を表す 3 つの頂点インデックスの配列。この配列は、 <i>numTriangles</i> *3 個の要素を持つ必要があります。
<i>numTriangles</i>	<i>inTriangles</i> 、および <i>outTriangles</i> 中の三角形の数
<i>userData</i>	キャッシュアルゴリズム用のチューニング係数を含む、 <a href="#">EdgeGeomKCacheOptimizerUserData</a> 構造体へのポインタ。
<i>outTriangles</i>	最適化された三角形リストが書き込まれる、割り当て済みの配列。 <i>inTriangles</i> と同じ数の要素を持つ必要があります。

## 返 り 値

なし。

## 解 説

この関数は、レンセリア科学技術大学の電気計算機システム工学部および数学部に所属する、Gang Lin、および Thomas P.-Y. Yu による「三次元メッシュレンダリングのための頂点キャッシュ法の改良 (An Improved Vertex Caching Scheme for 3D Mesh Rendering)」という論文に提示されたアルゴリズムの修正版です (URL については、「PlayStation®Edge ライブラリ 概要」の「関連ドキュメントとその他のリソース」セクションを参照してください)。修正点は、特に、ミニキャッシュへのヒット率を向上させるように各三角形の頂点を並べ替えたことと、頂点近傍の周囲で反時計回りの繰返しを行う代わりに、頂点近傍から最もコストの小さい三角形を検索するようにしたことです。どちらの修正も、テストモデル上では、著しい改善をもたらしました。

*m\_fifoMiss* と *m\_lruMissCost* との間の比は、このアルゴリズムが三角形順序を最適化するために行う方法に対して、大きな影響を与えます。*m\_lruMissCost* が *m\_fifoMiss* より著しく大きい場合、トライアングルリストは、ミニキャッシュからコヒーレンシーの高い三角形を選択するという短期的な利益については高度に最適化されますが、変換後のキャッシュ動作については低いパフォーマンスを示す可能性があります。同様に、*m\_fifoMiss* が相対的に大きい場合、頂点プログラムを再実行する必要はないにせよ、頂点を絶えずミニキャッシュにコピーし直さなくてはならないという、厳しい罰を受

ける可能性があります。最もバランスのよい結果を与える  $m\_fifoMiss:m\_lruMissCost$  の比は 4:1 です。

定数  $m\_k1$ 、 $m\_k2$ 、 $m\_k3$  は、論文で記述されたものとはならずしも厳密に一致していないことに注意してください。これは、潜在的な中心頂点の得点の合計は、変換後のキャッシュミスの数と厳密には一致しないからです。したがってこの論文に示されている定数値の 3 次元グラフは、Edge を利用するディベロッパが会合する可能性の高いものと同じではありません。このアルゴリズムでは、表面はそれほど滑らかではなく、いかなるモデルに対しても最良の結果を与えるようなはっきりした勾配が存在しないことも珍しくありません。これらの定数を変更することが、最適なトライアングルリストを実現するための唯一の方法です。

## **関 連 項 目**

[EdgeGeomKCacheOptimizerUserData](#)

---

# edgeGeomKCacheOptimizerHillclimber

---

定数k1,k2,k3 を自動的に変化させる [edgeGeomKCacheOptimizer](#) 用のフロントエンド

## 定 義

---

```
#include <libedgegeomtool.h>
void edgeGeomKCacheOptimizerHillclimber(
    const uint32_t *inTriangles,
    uint32_t numTriangles,
    void *userData,
    uint32_t *outTriangles
)
```

## 呼 出 条 件

---

スレッドから呼び出し可能。  
マルチスレッドセーフである。

## 引 数

---

<i>inTriangles</i>	三角形を表す3つの頂点インデックスの配列。この配列は、 <i>numTriangles</i> *3個の要素を持つ必要があります。
<i>numTriangles</i>	<i>inTriangles</i> および <i>outTriangles</i> 中の三角形の数
<i>userData</i>	kcacheアルゴリズム用のチューニング係数を含む、 <a href="#">EdgeGeomKCacheOptimizerHillclimberUserData</a> 構造体へのポインタ
<i>outTriangles</i>	最適化された三角形リストが書き込まれる、割り当て済みの配列。 <i>inTriangles</i> と同じ数の要素を持つ必要があります。

## 返 り 値

---

なし。

## 解 説

---

この関数は、頂点キャッシュオプティマイザの質を支配する定数空間に対して、指定された回数の調査を行います。この山登り法を利用した関数は、よりよい結果を求めて検索を改善するための方法として、定数面上にはっきりとよりよい結果を導くような勾配が存在しない場合、漸進的に刻み幅を小さくしながら、現在の定数の組み合わせから離れていきます。

## 参 照 項 目

---

[EdgeGeomKCacheOptimizerHillclimberUserData](#)

## コールバック関数



# EdgeGeomVertexCacheOptimizerFunc

分割された三角形リストを RSX®キャッシュ用に最適化するために、ライブラリによって呼び出されるコールバック関数

## 定 義

```
#include <libedgegeomtool.h>
typedef void (*EdgeGeomVertexCacheOptimizerFunc) (
    const uint32_t *triangles,
    uint32_t numTriangles,
    void *userData,
    uint32_t *outTriangles
);
```

## 引 数

<i>triangles</i>	頂点インデックスの入力配列。連続する3つのインデックスが三角形を表します。
<i>numTriangles</i>	三角形の数
<i>userData</i>	ユーザの指定するデータへのポインタ
<i>outTriangles</i>	最適化された三角形リストを出力するバッファへのポインタ。 このバッファのサイズは、 <code>3*numTriangles*sizeof(uint32_t)</code> です。

## 返 り 値

なし

## 解 説

このコールバック関数は、特定のパーティションの三角形リストを最適化する必要がある場合に、ライブラリによって呼び出されます。

ユーザは、[EdgeGeomPartitionerInput](#)構造体の中の *m\_cacheOptimizerFunc* や *m\_cacheOptimizerUserData* を介して、ユーザ独自の関数を登録したり、データポインタを指定したりすることができます。

Edgeでは、[edgeGeomKCacheOptimizer\(\)](#)、および [edgeGeomKCacheOptimizerHillclimber\(\)](#) という2つの最適化関数を提供しています。

## 関 連 項 目

[EdgeGeomPartitionerInput](#), [edgeGeomKCacheOptimizer](#), [edgeGeomKCacheOptimizerHillclimber](#)

---

# EdgeGeomCustomPartitionDataSizeFunc

---

入力パーティション用の追加のカスタムデータのサイズを返すコールバック関数

## 定 義

---

```
#include <libedgegeomtool.h>
typedef uint32_t (*EdgeGeomCustomPartitionDataSizeFunc) (
    const EdgeGeomPartitionElementCounts &partitionContents
);
```

## 引 数

---

*partitionContents* 現在のパーティションの説明

## 返 り 値

---

ユーザアプリケーションが入力パーティションのために必要とするデータのサイズ。

## 解 説

---

Edge パーティショナーは、パーティションにどの三角形を追加するべきかを選択する際に、この関数を呼び出します。

*partitionContents* は、現在構築中のパーティションの説明です。

この関数は、パーティションが必要とする（標準の Edge バッファ以外の）あらゆる追加データのサイズを（バイト単位で）返す必要があります。

## 注 意

---

この関数は、頻繁に呼び出される可能性があります。したがって、この関数で行う計算はなるべく単純なものにする必要があります。さもないと、この関数がすぐにツールの実行時間の大部分を占めることになります。

## 関 連 項 目

---

[EdgeGeomPartitionerInput](#), [EdgeGeomPartitionElementCounts](#)

# 列挙

# EdgeGeomIndexesFlavor

セグメントで使われるさまざまなインデックスデータの種類の表す定数

## 定 義

```
enum EdgeGeomIndexesFlavor
{
    kIndexesU16TriangleListCW =
        EDGE_GEOM_INDEXES_U16_TRIANGLE_LIST_CW,
    kIndexesU16TriangleListCCW =
        EDGE_GEOM_INDEXES_U16_TRIANGLE_LIST_CCW,
    kIndexesCompressedTriangleListCW =
        EDGE_GEOM_INDEXES_COMPRESSED_TRIANGLE_LIST_CW,
    kIndexesCompressedTriangleListCCW =
        EDGE_GEOM_INDEXES_COMPRESSED_TRIANGLE_LIST_CCW,
};
```

## 解 説

列挙型 値	解説
kIndexesU16TriangleListCW	時計回りの三角形順序をもつ16ビットのインデックスデータ
kIndexesU16TriangleListCCW	反時計回りの三角形順序をもつ16ビットのインデックスデータ
kIndexesCompressedTriangleListCW	時計回りの三角形順序をもつソフトウェア圧縮されたインデックスデータ
kIndexesCompressedTriangleListCCW	反時計回りの三角形順序をもつソフトウェア圧縮されたインデックスデータ

## 関 連 項 目

[EdgeGeomSegmentFormat](#)、[EdgeGeomKCacheOptimizerHillclimberUserData](#)、[EdgeGeomPartitionerInput](#)、[edgeGeomMakeSpuConfigInfo](#)、[edgeGeomMakeIndexBuffer](#)

# EdgeGeomSkinningFlavor

実行するさまざまな種類の行列パレットスキニングを表す定数

## 定 義

```
enum EdgeGeomSkinningFlavor
{
    kSkinNone                = EDGE_GEOM_SKIN_NONE,
    kSkinNoScaling           = EDGE_GEOM_SKIN_NO_SCALING,
    kSkinUniformScaling      = EDGE_GEOM_SKIN_UNIFORM_SCALING,
    kSkinNonUniformScaling   = EDGE_GEOM_SKIN_NON_UNIFORM_SCALING,
    kSkinSingleBoneNoScaling = EDGE_GEOM_SKIN_SINGLE_BONE_NO_SCALING,
    kSkinSingleBoneUniformScaling =
        EDGE_GEOM_SKIN_SINGLE_BONE_UNIFORM_SCALING,
    kSkinSingleBoneNonUniformScaling =
        EDGE_GEOM_SKIN_SINGLE_BONE_NON_UNIFORM_SCALING,
};
```

## 解 説

列挙型 値	解説
kSkinNone	スキニングを実行しません。
kSkinNoScaling	単位行列スキニング
kSkinUniformScaling	スキニングを実行します。
kSkinNonUniformScaling	スキニングを実行して、正規変換のための余因子行列を計算します。
kSkinSingleBoneNoScaling	単一ボーン単位行列スキニング
kSkinSingleBoneUniformScaling	単一ボーンスキニング
kSkinSingleBoneNonUniformScaling	単一ボーンスキニング。 正規変換のための余因子行列を計算します。

## 関 連 項 目

[EdgeGeomSegmentFormat](#)、[EdgeGeomPartitionerInput](#)、[edgeGeomMakeSpuConfigInfo](#)、[edgeGeomMakeSkinningBuffer](#)

# EdgeGeomMatrixFormat

サポートされたスキニング行列フォーマットを表す定数

## 定 義

```
enum EdgeGeomMatrixFormat
{
    kMatrix3x4RowMajor = EDGE_GEOM_MATRIX_3x4_ROW_MAJOR,
    kMatrix4x4RowMajor = EDGE_GEOM_MATRIX_4x4_ROW_MAJOR,
    kMatrix4x4ColumnMajor = EDGE_GEOM_MATRIX_4x4_COLUMN_MAJOR,
};
```

## 解 説

列挙型 値	解説
kMatrix3x4RowMajor	これは Edge のネイティブの行列型で、メモリの使用量に関して最も効率的です。この行列は、「DirectX スタイル」の 4x4 行列の上 3 行です。4 行目は常に [0, 0, 0, 1] なので、明示的に保存する必要はありません。
kMatrix4x4RowMajor	「DirectX スタイル」の行優先形式の 4x4 行列を指定します。この型が提供されるのは、主に便宜上の理由からです。4x4 行列は、3x4 行列より 33% 多くのメモリを使用することは言うまでもありません。
kMatrix4x4ColumnMajor	「OpenGL スタイル」の列優先形式の 4x4 行列を指定します。この型が提供されるのは、主に便宜上の理由からです。4x4 行列は、3x4 行列より 33% 多くのメモリを使用することは言うまでもありません。

## 関 連 項 目

[EdgeGeomSegmentFormat](#)、[EdgeGeomPartitionerInput](#)、[edgeGeomMakeSpuConfigInfo](#)、[edgeGeomMakeSkinningBuffer](#)

# EdgeGeomCullingFlavor

実行する三角形カリングテストのさまざまな組合せを表す定数

## 定 義

```
enum EdgeGeomCullingFlavor
{
    kCullNone                = EDGE_GEOM_CULL_NONE,
    kCullFrustum             = EDGE_GEOM_CULL_FRUSTUM,
    kCullBackFacesAndFrustum = EDGE_GEOM_CULL_BACKFACES_AND_FRUSTUM,
    kCullFrontFacesAndFrustum = EDGE_GEOM_CULL_FRONTFACES_AND_FRUSTUM,
};
```

## 解 説

列挙型 値	解説
kCullNone	三角形カリングを実行しません。
kCullFrustum	フラスタムおよびピクセル中心三角形カリングを実行します。
kCullBackFacesAndFrustum	裏面、フラスタム、およびピクセル中心三角形カリングを実行します。
kCullFrontFacesAndFrustum	表面、フラスタム、およびピクセル中心三角形カリングを実行します。

## 関 連 項 目

[EdgeGeomSegmentFormat](#)、[EdgeGeomPartitionerInput](#)、[edgeGeomGetScratchBufferSizeInQwords](#)