

PlayStation®Edge Animation

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
PPU/SPU共有実行時データ型.....	6
EdgeAnimPpuContext	7
EdgeAnimJointTransform	8
EdgeAnimBlendBranch	9
EdgeAnimBlendLeaf	11
EdgeAnimSkeleton	12
EdgeAnimAnimation	14
EdgeAnimMirrorPair	16
EdgeAnimCustomDataTable	17
SPU実行時データ型	18
EdgeAnimSpuContext	19
EdgeAnimFrameSetInfo	20
EdgeAnimPoseInfo	21
EdgeAnimCommand	22
PPU/SPU共有関数	23
edgeAnimSkeletonGetJointIndexByName	24
edgeAnimSkeletonGetJointIndexByHash	25
edgeAnimSkeletonGetUserChannelIndexByName	26
edgeAnimSkeletonGetUserChannelIndexByHash	27
edgeAnimGenerateNameHash	28
edgeAnimGetAnimTag	29
edgeAnimGetSkelTag	30
PPU関数	31
edgeAnimPpuInitialize	32
edgeAnimPpuFinalize	34
edgeAnimComputeExternalStorageSize	35
edgeAnimEvaluateJoint	36
edgeAnimEvaluateUserChannel	37
SPU関数	38
edgeAnimSpuInitialize	39
edgeAnimSpuFinalize	40
edgeAnimProcessBlendTree	41
edgeAnimProcessCommandList	43
edgeAnimLocalJointsToWorldMatrices3x4	44
edgeAnimLocalJointsToWorldMatrices4x4	45
edgeAnimLocalJointsToWorldJoints	46
edgeAnimWorldJointsToLocalJoints	47
edgeAnimJointsToMatrices4x4	48
edgeAnimMatrices4x4ToJoints	49
edgeAnimJointsToMatrices3x4	50

edgeAnimMatrices3x4ToJoints	51
edgeAnimBlendJointsLinear	52
edgeAnimBlendJointsRelative	54
edgeAnimBlendPose.....	56
edgeAnimBlendUserLinear	57
edgeAnimBlendUserRelative	59
edgeAnimPoseStackPush.....	61
edgeAnimPoseStackPop.....	62
edgeAnimPoseStackGetPose	63
edgeAnimCustomDataChunk.....	64
コールバック関数.....	65
EdgeAnimLeafCallback.....	66
EdgeAnimBranchCallback	68
EdgeAnimUserCallback	69
定数	71
EdgeAnimBlendOp.....	72
EdgeAnimRelativeBlendMode	73

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 ライブラリのアニメーションコンポーネントの API リファレンスです。アニメーションコンポーネントは、SPU を利用して、アニメーションジョイントツリーのブレンディングや変換を効率的に行うために利用します。

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開発しようとしている PlayStation®3 ディベロッパーのため書かれたものです。ディベロッパーは、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 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」
- 「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 LZO 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、そして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。
青字 + 下線	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）。

PPU/SPU共有実行時データ型

EdgeAnimPpuContext

メインメモリ上にある各 SPU 用の一時記憶領域に関する情報が含まれている

定 義

```
#include <edge/anim/edgeanim_structs.h>
typedef struct
{
    /* ドキュメント化されていません */
} EdgeAnimPpuContext ;
```

解 説

この構造体には、メインメモリ上にある各SPU用の一時記憶領域に関する情報が含まれています。アプリケーションは、この構造体に領域を割り当て、[edgeAnimPpuInitialize\(\)](#)に渡して初期化する必要があります。

注 意

この構造体は SPU によってアクセスされるので、ヒープに割り当てるか、もしくはグローバル領域に存在する必要があります。

関 連 項 目

[edgeAnimPpuInitialize](#)

EdgeAnimJointTransform

単一のジョイント変換を記述する

定 義

```
#include <edge/anim/edgeanim_structs.h>
typedef struct
{
    Vectormath::Aos::Quat rotation;
    Vectormath::Aos::Point3 translation;
    Vectormath::Aos::Vector4 scale;
} EdgeAnimJointTransform ;
```

メ ン バ

<i>rotation</i>	変換のうちの回転部分
<i>translation</i>	変換のうちの平行移動部分
<i>scale</i>	変換のうちの拡大縮小部分

解 説

この構造体は、特定のジョイント変換を記述します。

EdgeAnimBlendBranch

ブレンドツリー中のブランチを指定する

定 義

```
#include <edge/anim/edgeanim_structs.h>
typedef struct
{
    uint16_t operation;
    uint16_t left;
    uint16_t right;
    uint16_t flags;
    float alpha;
    uint32_t userVal;
} EdgeAnimBlendBranch;
```

メ ン バ

<i>operation</i>	ブレンド操作 (下記参照)
<i>left</i>	左のリーフ/ブランチのインデックス (下記参照)
<i>right</i>	右のリーフ/ブランチのインデックス (下記参照)
<i>flags</i>	ブランチフラグ (下記を参照)
<i>alpha</i>	ブレンド係数 (0~1)
<i>userVal</i>	ユーザ定義の値

Operation メンバは、以下の値を取ることができます。

マクロ	値	解説
EDGE_ANIM_BLENDOP_BLEND_LINEAR	0	線形ブレンド
EDGE_ANIM_BLENDOP_BLEND_ADD_DELTA_RIGHT	1	左 ($\alpha = 0.0$) と左+右 ($\alpha = 1.0$) との間の線形ブレンド
EDGE_ANIM_BLENDOP_BLEND_ADD_DELTA_LEFT	2	右 ($\alpha = 0.0$) と右+左 ($\alpha = 1.0$) との間の線形ブレンド
EDGE_ANIM_BLENDOP_COMPOSE_ADD_RIGHT	3	左 + 右を合成します
EDGE_ANIM_BLENDOP_COMPOSE_ADD_LEFT	4	右 + 左を合成します
EDGE_ANIM_BLENDOP_COMPOSE_SUB_RIGHT_FROM_LEFT	5	左 - 右を合成します
EDGE_ANIM_BLENDOP_COMPOSE_SUB_LEFT_FROM_RIGHT	6	右 - 左を合成します

left と *right* には、以下の2つのフラグのいずれかとインデックスとの間でビット OR をとった値を指定します。

マクロ	値	解説
EDGE_ANIM_BLEND_TREE_INDEX_LEAF	0x8000	リーフ指定子
EDGE_ANIM_BLEND_TREE_INDEX_BRANCH	0x4000	ブランチ指定子

flags には、以下の値のいずれかでビット OR をとった値を指定できます。

マクロ	値	解説
EDGE_ANIM_FLAG_MIRROR	1	ブレンドの結果がミラーされる。

解 説

この構造体は、ブレンドツリー中のブランチを指定するために使われます。ブランチは、2つのツリー要素の間のブレンド操作を表します。

左右のインデックスは、それぞれリーフもしくはブランチのいずれかを表し、ブレンドツリー中のリーフやブランチの配列の要素に対応しています。どちらの場合も、インデックス 0 が配列の最初の要素です。

`userVal`には、オプションで、ユーザ定義の値を渡すことができます。この値は、ユーザリーフ処理用のリーフコールバック関数 [EdgeAnimLeafCallback\(\)](#) に（もしあれば）渡されます。

関 連 項 目

[edgeAnimProcessBlendTree](#)、[edgeAnimBlendPose](#)、[EdgeAnimLeafCallback](#)

EdgeAnimBlendLeaf

ブレンドツリー中のリーフを指定する

定 義

```
#include <edge/anim/edgeanim_structs.h>
typedef struct
{
    union
    {
        uint32_t animationHeaderEa;
        uint32_t poseAddr;
    }
    uint16_t animationHeaderSize;
    uint16_t flags;
    float evalTime;
    uint32_t userVal;
} EdgeAnimBlendLeaf;
```

メ ン バ

<i>animationHeaderEa</i>	アニメーションヘッダの実効アドレス
<i>poseAddr</i>	計算済みのポーズのアドレス
<i>animationHeaderSize</i>	アニメーションヘッダのサイズ
<i>flags</i>	リーフフラグ（下記を参照）
<i>evalTime</i>	評価時間（秒）
<i>userVal</i>	ユーザ定義の値

解 説

この構造体は、ブレンドツリー中のリーフを指定します。リーフは、特定の時刻において評価されるアニメーションを表します。

evalTime は、評価時に（0、d）の範囲に強制的に補正されます。dはクリップの持続時間です。

*userVal*には、オプションでユーザ定義の値を渡すことができます。この値は、ユーザリーフ処理用のリーフコールバック関数 [EdgeAnimLeafCallback\(\)](#) に（もしあれば）渡されます。

葉が計算済みのポーズを表している場合、*poseAddr* はメインメモリもしくはローカルストア中のポーズのアドレスを指定するために使われます。この場合には、EDGE_ANIM_FLAG_POSE_FROM_MAIN、もしくはEDGE_ANIM_FLAG_POSE_FROM_LOCAL フラグをそれぞれ設定する必要があります。

flags には、以下の値のいずれかでビット OR をとった値を指定できます。

マクロ	値	解説
EDGE_ANIM_FLAG_MIRROR	1	アニメーションがミラーされる
EDGE_ANIM_FLAG_POSE_FROM_MAIN	2	葉はメインメモリ中の計算済みのポーズを指定する
EDGE_ANIM_FLAG_POSE_FROM_LOCAL	4	葉はローカルメモリ中の計算済みのポーズを指定する

関 連 項 目

[edgeAnimProcessBlendTree](#)、[EdgeAnimLeafCallback](#)

EdgeAnimSkeleton

スケルトンのバイナリヘッダを記述する

定 義

```
#include <edge/anim/edgeanim_structs.h>
typedef struct
{
    uint32_t tag;
    uint32_t sizeTotal;
    uint32_t sizeCustomData;
    uint32_t sizeNameHashes;
    uint16_t numJoints;
    uint16_t numUserChannels;
    uint16_t numSimdHierarchyQuads;
    uint16_t locomotionJointIndex;
    uint32_t offsetBasePose;
    uint32_t offsetParentIndicesArray;
    uint32_t offsetJointNameHashArray;
    uint32_t offsetUserChannelNameHashArray;
    uint32_t offsetUserChannelNodeNameHashArray;
    uint32_t offsetUserChannelFlagsArray;
    uint32_t offsetCustomData;
    uint32_t pad1[3];
    uint16_t simdHierarchy[0];
} EdgeAnimSkeleton;
```

メ ン バ

<i>tag</i>	バージョンタグ
<i>sizeTotal</i>	合計サイズ
<i>sizeCustomData</i>	カスタムユーザデータまたはカスタムデータテーブルのサイズ
<i>sizeNameHashes</i>	名前ハッシュのサイズ
<i>numJoints</i>	ジョイントの数
<i>numUserChannels</i>	ユーザチャンネルの数
<i>numSimdHierarchyQuads</i>	単一命令多重データ (SIMD) 階層中のクワッドの数
<i>locomotionJointIndex</i>	移動差分に使用されるジョイントのインデックス
<i>offsetBasePose</i>	ベースポーズへのオフセット (EdgeAnimJointTransform*)
<i>offsetParentIndicesArray</i>	親インデックス配列へのオフセット (int16_t*)
<i>offsetJointNameHashArray</i>	ジョイント名ハッシュ配列へのオフセット (uint32_t*)
<i>offsetUserChannelNameHashArray</i>	ユーザチャンネル名ハッシュ配列へのオフセット (uint32_t*)
<i>offsetUserChannelNodeNameHashArray</i>	ユーザチャンネルノード名ハッシュ配列 (uint32_t*)へのオフセット
<i>offsetUserChannelFlagsArray</i>	ユーザチャンネルフラグ配列 (uint8_t*) へのオフセット
<i>offsetCustomData</i>	カスタムユーザデータへのオフセット
<i>pad1[3]</i>	未使用
<i>simdHierarchy[0]</i>	SIMD 階層

解 説

この構造体は、スケルトンのバイナリヘッダを記述します。

オフセットは、すべてアドレスからの相対値です。オフセットからポインタを取得するには、`EDGE_OFFSET_GET_POINTER()` マクロを使ってください。

`tag`にはバイナリの現在のバージョン番号が含まれており、[edgeAnimGetSkelTag\(\)](#)が返す値と一致する必要があります。

`sizeTotal`はバイナリの合計サイズを提供します。カスタムデータとハッシュ名はスケルトンヘッダの末尾に存在しているため、アプリケーションでそれらの情報が不要であれば、それらのアップロードをスキップしてもかまいません。その場合、それらのサイズを合計サイズから差し引きます。（コア SPU 処理関数ではカスタムデータやハッシュ名は一切必要ありません。）

`offsetParentIndicesArray`には、ジョイントの親インデックスの配列へのオフセットが含まれています（-1 は親が存在しないことを示す）。

`simdHierarchy`には、SPU 実行時に必要な SIMD に適した形式の階層情報が含まれています。`uint32_t`型の要素には、それぞれ 16 ビットの親と子のインデックスのペアが入っています。この配列では、要素が重複していてもかまいません。

`numSimdHierarchyQuads`には、`simdHierarchy`のクワッドワードサイズを指定します。

関 連 項 目

[edgeAnimGetSkelTag](#)

EdgeAnimAnimation

アニメーションのバイナリヘッダを記述する

定 義

```
#include <edge/anim/edgeanim_structs.h>
typedef struct
{
    uint32_t tag;
    float duration;
    float sampleFrequency;
    uint16_t sizeHeader;
    uint16_t numJoints;
    uint16_t numFrames;
    uint16_t numFrameSets;
    uint16_t evalBufferSizeRequired;
    uint16_t numConstRChannels;
    uint16_t numConstTChannels;
    uint16_t numConstSChannels;
    uint16_t numConstUserChannels;
    uint16_t numAnimRChannels;
    uint16_t numAnimTChannels;
    uint16_t numAnimSChannels;
    uint16_t numAnimUserChannels;
    uint16_t flags;
    uint32_t sizeJointsWeightArray;
    uint32_t eaUserJointWeightArray;
    uint32_t offsetJointsWeightArray;
    uint32_t offsetFrameSetDmaArray;
    uint32_t offsetFrameSetInfoArray;
    uint32_t offsetConstRData;
    uint32_t offsetConstTData;
    uint32_t offsetConstSData;
    uint32_t offsetConstUserData;
    uint32_t offsetPackingSpecs;
    uint32_t offsetCustomData;
    uint32_t sizeCustomData;
    uint32_t offsetLocomotionDelta;
    uint32_t pad1;
    uint16_t channelTables[0];
} EdgeAnimAnimation;
```

メ ン バ

<i>tag</i>	バージョンタグ
<i>duration</i>	アニメーションの持続時間 (秒)
<i>sampleFrequency</i>	サンプリング周波数 (Hz)
<i>sizeHeader</i>	ヘッダのサイズ (16 バイトアラインメント)
<i>numJoints</i>	ジョイントの数
<i>numFrames</i>	フレームの数
<i>numFrameSets</i>	フレームセットの数
<i>evalBufferSizeRequired</i>	必要な評価バッファのサイズ
<i>numConstRChannels</i>	定数回転チャンネルの数
<i>numConstTChannels</i>	定数平行移動チャンネルの数
<i>numConstSChannels</i>	定数拡大縮小チャンネルの数
<i>numConstUserChannels</i>	定数ユーザチャンネルの数

<i>numAnimRChannels</i>	アニメーション回転チャンネルの数
<i>numAnimTChannels</i>	アニメーション平行移動チャンネルの数
<i>numAnimSChannels</i>	アニメーション拡大縮小チャンネルの数
<i>numAnimUserChannels</i>	アニメーションユーザチャンネルの数
<i>flags</i>	内部フラグ
<i>sizeJointsWeightArray</i>	重み配列のサイズ（配列が存在しない場合は 0）
<i>eaUserJointWeightArray</i>	ジョイント重み配列用のユーザオーバーライド。ツールで NULL に設定する必要がある
<i>offsetJointsWeightArray</i>	重み配列へのオフセット
<i>offsetFrameSetDmaArray</i>	フレームセット DMA 配列 (CellDmaListElement*) へのオフセット
<i>offsetFrameSetInfoArray</i>	フレームセット情報配列 (EdgeAnimFrameSetInfo*) へのオフセット
<i>offsetConstRData</i>	定数回転データへのオフセット
<i>offsetConstTData</i>	定数平行移動データへのオフセット
<i>offsetConstSData</i>	定数拡大縮小データへのオフセット
<i>offsetConstUserData</i>	定数ユーザデータへのオフセット
<i>offsetPackingSpecs</i>	パック指定へのオフセット（ビットパックチャンネル用）
<i>offsetCustomData</i>	カスタムユーザデータまたはカスタムデータテーブルへのオフセット
<i>sizeCustomData</i>	カスタムユーザデータまたはカスタムデータテーブルのサイズ
<i>offsetLocomotionDelta</i>	オプションの移動差分のオフセット（差分がない場合には 0）
<i>pad1</i>	未使用
<i>channelTables</i>	チャンネルテーブル

解 説

この構造体は、アニメーションのバイナリヘッダを記述します。

オフセットは、すべてアドレスからの相対値です。オフセットからポインタを取得するには、`EDGE_OFFSET_GET_POINTER()` マクロを使ってください。

*tag*にはバイナリの現在のバージョン番号が含まれており、[edgeAnimGetAnimTag\(\)](#) が返す値に一致する必要があります。

*evalBufferSizeRequired*には、クリップに必要な評価バッファのサイズを指定します。[edgeAnimSpuInitialize\(\)](#) の *maxSizeEvalBuffer* 引数に渡す値は、このサイズ以上である必要があります。

*sizeJointsWeightArray*には、各ジョイントの重み配列のサイズを指定します。このサイズが 0 の場合には、重みがすべて 1 に設定されたフルボディアニメーションであると仮定されます。

関 連 項 目

[edgeAnimGetAnimTag](#)

EdgeAnimMirrorPair

ジョイントまたはジョイントペアに対するミラー操作を指定する

定 義

```
#include <edge/anim/edgeanim_structs.h>
typedef struct
{
    uint16_t idx0;
    uint16_t idx1;
    uint32_t spec;
} EdgeAnimMirrorPair;
```

メ ン バ

<i>idx0</i>	1 番目のジョイントのインデックス
<i>idx1</i>	2 番目のジョイントのインデックス
<i>spec</i>	ミラーリング処理

解 説

この構造体はジョイントまたはジョイントペアのミラーリング操作を指定する目的に使用します。
*spec*は*idx0*および*idx1*の2つのジョイントに対して行われるミラーリング操作を定義するもので、一連の符号反転と、回転コンポーネントおよび平行移動コンポーネントのコンポーネント入れ替えで構成されています。2つのジョイントは、ポーズ内でエントリの交換が行われます。
 この構造体は単一（非ペア）ジョイントのミラーリングの指定にも使用され、この場合には*idx0*と*idx1*が同一になります。
*spec*には、以下の値を指定することができます。

マクロ	値	解説
EDGE_ANIM_MIRROR_SPEC_NON_PAIRED	0x881122b3U	非ペアジョイント用のミラーリング操作
EDGE_ANIM_MIRROR_SPEC_LINK	0xb8219203U	非ペアの親を持つペアジョイント用のミラーリング操作
EDGE_ANIM_MIRROR_SPEC_PAIRED	0x08192233U	ペアの親を持つペアジョイント用のミラーリング操作

関 連 項 目

[edgeAnimProcessCommandList](#)、[edgeAnimProcessBlendTree](#)

EdgeAnimCustomDataTable

複数の独立したカスタムデータエントリを格納するためのオプション構造体を記述する。このデータは [EdgeAnimSkeleton](#) と [EdgeAnimAnimation](#) で使用され、ハッシュ名識別子を使ってアクセス可能となっている。

定 義

```
#include <edge/anim/edgeanim_structs.h>
typedef struct
{
    uint32_t numEntries;
    uint32_t offsetHashArray;
    uint32_t offsetEntrySizes;
    uint32_t offsetEntries;
} EdgeAnimCustomDataTable ;
```

メ ン バ

<i>numEntries</i>	テーブル内のデータエントリの数
<i>offsetHashArray</i>	ハッシュ名リスト (uint32_t []) へのオフセット
<i>offsetEntrySizes</i>	エントリサイズリスト (uint32_t []) へのオフセット
<i>offsetEntries</i>	エントリデータリストへのオフセット

解 説

この構造体は、[EdgeAnimSkeleton](#)・[EdgeAnimAnimation](#) 構造体で使用するための複数の独立したカスタムデータを格納する方法を提供します。このテーブルの使用はオプションで、Edge Animation でカスタムデータをサポートするための基盤として実装されたものです。

各エントリの識別は、[edgeAnimGenerateNameHash\(\)](#) で生成されるハッシュ値を使って行います。テーブル内のエントリを検索するためのヘルパー関数 [edgeAnimCustomDataChunk\(\)](#) が利用可能となっています。

関 連 項 目

[edgeAnimGenerateNameHash](#)、[edgeAnimCustomDataChunk](#)

SPU実行時データ型

EdgeAnimSpuContext

SPU のコンテキスト固有のデータが含まれる構造体

定 義

```
#include <edge/anim/edgeanim_structs.h>
typedef struct
{
    /* ドキュメント化されていません */
} EdgeAnimSpuContext ;
```

解 説

この構造体には、SPUのコンテキスト固有のデータが含まれています。アプリケーションは、この構造体に領域を割り当て、[edgeAnimSpuInitialize\(\)](#)に渡して初期化する必要があります。

関 連 項 目

[edgeAnimSpuInitialize](#)

EdgeAnimFrameSetInfo

アニメーション中の特定のフレームセット（フレームの部分集合）を記述する構造体

定 義

```
#include <edge/anim/edgeanim_structs.h>
typedef struct
{
    uint16_t baseFrame;
    uint16_t numIntraFrames;
} EdgeAnimFrameSetInfo;
```

メ ン バ

<i>baseFrame</i>	フレームセット中の最初のフレームのインデックス
<i>numIntraFrames</i>	フレームセット中のイントラフレームの数

解 説

この構造体は、アニメーション中の特定のフレームセット（フレームの部分集合）を記述します。

EdgeAnimPoseInfo

単一のポーズに関する情報が含まれる構造体

定 義

```
#include <edge/anim/edgeanim_structs.h>
typedef struct
{
    EdgeAnimJointTransform* jointArray;
    uint8_t* weightArray;
    float* userChannelArray;
    uint8_t* userChannelWeightArray;
    uint32_t* flags;
    uint32_t pad;
} EdgeAnimPoseInfo;
```

メ ン バ

<i>jointArray</i>	ジョイント配列へのポインタ
<i>weightArray</i>	重み配列へのポインタ
<i>userChannelArray</i>	ユーザチャンネル配列へのポインタ
<i>userChannelWeightArray</i>	ユーザチャンネル重み配列へのポインタ
<i>flags</i>	フラグへのポインタ
<i>pad</i>	未使用

解 説

この構造体には、単一のポーズに関する情報が含まれています。

関 連 項 目

[edgeAnimPoseStackGetPose](#)

EdgeAnimCommand

単一のアニメーションコマンドを記述する構造体

定 義

```
#include <edge/anim/edgeanim_structs.h>
typedef struct
{
    uint16_t command;
    uint16_t arg0;
    union{
        const EdgeAnimBlendLeaf* leaf;
        const EdgeAnimBlendBranch* branch;
        uint32_t user;
    };
} EdgeAnimCommand;
```

メ ン バ

command コマンド ID (下記参照)
arg0 ユーザコマンドの追加引数
leaf コマンドが `EDGE_ANIM_CMD_PUSH_AND_EVAL` である場合のリーフへのポインタ
branch コマンドが `EDGE_ANIM_CMD_BLEND_AND_POP` である場合のブランチへのポインタ
user カスタムコマンド用のユーザ指定値

command メンバは、以下の値に設定することができます。

マクロ	値	解説
<code>EDGE_ANIM_CMD_END_LIST</code>	0	コマンドリストの最後
<code>EDGE_ANIM_CMD_EVAL</code>	1	ポーズスタックの一番上を評価する
<code>EDGE_ANIM_CMD_PUSH_AND_EVAL</code>	2	一時的なポーズを割り当てて評価する
<code>EDGE_ANIM_CMD_BLEND_AND_POP</code>	3	ブレンドを実行してポップする
<code>EDGE_ANIM_CMD_MIRROR</code>	4	ポーズスタック最上位のミラーポーズ

解 説

この構造体は、単一のアニメーションコマンドを記述します。コマンドは、ブレンドツリーを処理する際に、Edge Animation によって内部的に生成されます。

ユーザの生成したコマンドリストの場合、上記以外のコマンドはすべてユーザコマンドとして解釈され、処理は [edgeAnimProcessCommandList](#) からユーザ指定のコールバックに渡されます。

関 連 項 目

[edgeAnimProcessCommandList](#)、[edgeAnimProcessBlendTree](#)

PPU/SPU共有関数

edgeAnimSkeletonGetJointIndexByName

指定された名前をもつジョイントのインデックスを返す

定 義

```
#include <edge/anim/edgeanim_spu.h>
あるいは、
#include <edge/anim/edgeanim_ppu.h>

int32_t edgeAnimSkeletonGetJointIndexByName (
    const EdgeAnimSkeleton* skeleton,
    const char* jointName
)
```

呼 出 条 件

マルチスレッドセーフである。

引 数

<i>skeleton</i>	ジョイントを含むスケルトンへのポインタ
<i>jointName</i>	ジョイントの名前

返 り 値

ジョイントのインデックスを返します。
ジョイントが見からない場合には-1 を返します。

解 説

この関数は、指定された名前をもつジョイントのインデックスを返します。

注 意

この関数は、内部的に文字列のハッシュ値を計算します。したがって、この関数よりも、計算済みのハッシュ値を渡して [edgeAnimSkeletonGetJointIndexByHash\(\)](#) を呼び出す方がパフォーマンス的には優れています。

関 連 項 目

[edgeAnimSkeletonGetJointIndexByHash](#)

edgeAnimSkeletonGetJointIndexByHash

指定されたハッシュ値の名前をもつジョイントのインデックスを返す

定 義

```
#include <edge/anim/edgeanim_spu.h>
あるいは、
#include <edge/anim/edgeanim_ppu.h>

int32_t edgeAnimSkeletonGetJointIndexByHash (
    const EdgeAnimSkeleton* skeleton,
    uint32_t jointHash
)
```

呼 出 条 件

マルチスレッドセーフである。

引 数

<i>skeleton</i>	ジョイントを含むスケルトンへのポインタ
<i>jointHash</i>	ジョイント名のハッシュ値

返 り 値

ジョイントのインデックスを返します。
ジョイントが見からない場合には-1 を返します。

解 説

この関数は、指定されたハッシュ値の名前をもつジョイントのインデックスを返します。

備 考

[edgeAnimGenerateNameHash\(\)](#) を呼び出すと、文字列からハッシュ値を生成することができます。

関 連 項 目

[edgeAnimSkeletonGetJointIndexByName](#)、[edgeAnimGenerateNameHash](#)

edgeAnimSkeletonGetUserChannelIndexByName

指定されたノード/名前の組を持つユーザチャンネルのインデックスを返す

定 義

```
#include <edge/anim/edgeanim_spu.h>
あるいは、
#include <edge/anim/edgeanim_ppu.h>

int32_t edgeAnimSkeletonGetUserChannelIndexByName (
    const EdgeAnimSkeleton* skeleton,
    const char* userChannelNodeName,
    const char* userChannelName
)
```

呼 出 条 件

マルチスレッドセーフである。

引 数

<i>skeleton</i>	ユーザチャンネルを含むスケルトンへのポインタ
<i>userChannelNodeName</i>	ユーザチャンネルノードの名前
<i>userChannelName</i>	ユーザチャンネルの名前

返 り 値

ユーザチャンネルのインデックスを返します。
ユーザチャンネルが見からない場合には-1を返します。

解 説

指定されたノード/名前の組を持つユーザチャンネルのインデックスを返します。

userChannelNodeName には、チャンネルの親ノードの名前を指定します。

userChannelName には、チャンネル自体の名前を指定します。

備 考

この関数は、内部的に文字列のハッシュ値を計算します。したがって、この関数よりも、計算済みのハッシュ値を渡して [edgeAnimSkeletonGetUserChannelIndexByHash\(\)](#) を呼び出すほうが、パフォーマンス的には優れています。

関 連 項 目

[edgeAnimSkeletonGetUserChannelIndexByHash](#)

edgeAnimSkeletonGetUserChannelIndexByHash

指定されたノード/名前の組を持つユーザチャンネルのインデックスを返す

定 義

```
#include <edge/anim/edgeanim_spu.h>
あるいは、
#include <edge/anim/edgeanim_ppu.h>

int32_t edgeAnimSkeletonGetUserChannelIndexByName (
    const EdgeAnimSkeleton* skeleton,
    const char* userChannelNodeName,
    const char* userChannelName
)
```

呼 出 条 件

マルチスレッドセーフである。

引 数

<i>skeleton</i>	ユーザチャンネルを含むスケルトンへのポインタ
<i>userChannelNodeNameHash</i>	ユーザチャンネルノードのハッシュ値
<i>userChannelNameHash</i>	ユーザチャンネルのハッシュ値

返 り 値

ユーザチャンネルのインデックスを返します。

ユーザチャンネルが見からない場合には-1 を返します。

解 説

指定されたノード/名前の組を持つユーザチャンネルのインデックスを返します。

userChannelNodeName には、チャンネルの親ノードのハッシュ名を指定します。

userChannelName には、チャンネル自体のハッシュ名を指定します。

備 考

[edgeAnimGenerateNameHash\(\)](#) を呼び出すと、文字列からハッシュ値を生成することができます。

関 連 項 目

[edgeAnimSkeletonGetUserChannelIndexByName](#)、[edgeAnimGenerateNameHash](#)

edgeAnimGenerateNameHash

文字列のハッシュ値を計算する

定 義

```
#include <edge/anim/edgeanim_spu.h>
あるいは、
#include <edge/anim/edgeanim_ppu.h>

uint32_t edgeAnimGenerateNameHash(
    const char* name
)
```

呼 出 条 件

マルチスレッドセーフである。

引 数

<i>name</i>	評価される文字列
-------------	----------

返 り 値

ハッシュ値を返します。

解 説

この関数は、文字列のハッシュ値を計算します。

edgeAnimGetAnimTag

アニメーションのバイナリタグを取得する

定 義

```
#include <edge/anim/edgeanim_common.h>
uint32_t edgeAnimGetAnimTag()
```

呼 出 条 件

マルチスレッドセーフである。

引 数

なし

返 り 値

アニメーションのバイナリタグを返します。

解 説

この関数は、[EdgeAnimAnimation](#)のバイナリタグを取得します。このタグは「EAnn」という形式で、EAは「EdgeAnimation」を意味し、*nn*は現在のバージョン番号になります。このバージョン番号は、フォーマットが変更されるたびにインクリメントされます。

関 連 項 目

[EdgeAnimAnimation](#)

edgeAnimGetSkelTag

スケルトンのバイナリタグを取得する

定 義

```
#include <edge/anim/edgeanim_common.h>
uint32_t edgeAnimGetSkelTag()
```

呼 出 条 件

マルチスレッドセーフである。

引 数

なし

返 り 値

スケルトンのバイナリタグを返します。

解 説

この関数は、[EdgeAnimSkeleton](#)のバイナリタグを取得します。このタグは、「ESnn」という形式で、ESは「EdgeSkeleton」を意味し、nnは現在のバージョン番号になります。このバージョン番号は、フォーマットが変更されるたびにインクリメントされます。

関 連 項 目

[EdgeAnimSkeleton](#)

PPU関数

edgeAnimPpuInitialize

PPU Edge Animation ライブラリを初期化する

定 義

```
#include <edge/anim/edgeanim_ppu.h>
void edgeAnimPpuInitialize(
    EdgeAnimPpuContext* ppuContext,
    uint32_t numSpus,
    int32_t spuExternalStorageMask,
    size_t sizeExternalStoragePerSpu,
    void* externalStorageBlock
)
```

呼 出 条 件

マルチスレッドセーフである（異なるスレッドが異なる ppuContexts で動作している場合）。

引 数

<i>ppuContext</i>	この関数によって書き込まれる PPU コンテキスト構造体のアドレス
<i>numSpus</i>	アニメーションを処理する SPU の最大数 (0~6)
<i>spuExternalStorageMask</i>	どの SPU が外部記憶を持っているかを指定するビットマスク
<i>sizeExternalStoragePerSpu</i>	各 SPU 用の外部記憶のサイズ
<i>externalStorageBlock</i>	外部記憶ブロックのアドレス

返 り 値

なし

解 説

この関数は、PPU Edge Animation ライブラリを初期化します。

ポーズスタックの外部記憶領域を必要としない場合には、最後の 3 つの引数は 0 に設定する必要があります。

ポーズスタックの外部記憶領域が必要な場合には、アプリケーションは、メインメモリからそれ用の領域を割り当てて指定する必要があります。このバッファは、16 バイト境界にアラインメントされている必要があります、そのサイズは、[edgeAnimComputeExternalStorageSize\(\)](#) の引数として、*numSpus*、*spuExternalStorageMask*、*sizeExternalStoragePerSpu* を渡すことによって求められます。

spuExternalStorageMask では、外部記憶を持つ SPU に対応する各ビットに 1 を指定します。

注 意

ppuContext 構造体は SPU によってアクセスされるので、ヒープに割り当てるか、もしくはグローバル領域に存在する必要があります。

関 連 項 目

[edgeAnimComputeExternalStorageSize](#)

edgeAnimPpuFinalize

PPU Edge Animation ライブラリの終了処理を行う

定 義

```
#include <edge/anim/edgeanim_ppu.h>
void edgeAnimPpuFinalize(
    EdgeAnimPpuContext* ppuContext
)
```

呼 出 条 件

マルチスレッドセーフである（異なるスレッドが異なる ppuContexts で動作している場合）。

引 数

<i>ppuContext</i>	PPU コンテキストへのポインタ
-------------------	------------------

返 り 値

なし

解 説

この関数は、PPU Edge Animation ライブラリの終了処理を行います。

edgeAnimComputeExternalStorageSize

ポーズキャッシュバッファの外部記憶領域の大きさを計算する

定 義

```
#include <edge/anim/edgeanim_ppu.h>
size_t edgeAnimComputeExternalStorageSize(
    uint32_t numSpus,
    int32_t spuExternalStorageMask,
    size_t sizeExternalStoragePerSpu
)
```

呼 出 条 件

マルチスレッドセーフである。

引 数

<i>numSpus</i>	アニメーションを処理する SPU の最大数 (0~6)
<i>spuExternalStorageMask</i>	どの SPU が外部記憶を持っているかを指定するビットマスク
<i>sizeExternalStoragePerSpu</i>	各 SPU 用の外部記憶のサイズ

返 り 値

計算された記憶領域のサイズを返します。

解 説

この関数は、ポーズキャッシュバッファの外部記憶領域の大きさを計算します。

この関数の引数は、[edgeAnimPpuInitialize\(\)](#) に渡すものと同じである必要があります。

関 連 項 目

[edgeAnimPpuInitialize](#)

edgeAnimEvaluateJoint

指定されたアニメーションおよび時刻において特定のジョイントを評価する

定 義

```
#include <edge/anim/edgeanim_ppu.h>
void edgeAnimEvaluateJoint(
    EdgeAnimJointTransform* outputJoint,
    const EdgeAnimAnimation* anim,
    const EdgeAnimSkeleton* skel,
    uint32_t jointIndex,
    float evalTime
)
```

呼 出 条 件

マルチスレッドセーフである。

引 数

<i>outputJoint</i>	出力ジョイント変換へのポインタ
<i>anim</i>	アニメーションへのポインタ
<i>skel</i>	スケルトンへのポインタ
<i>jointIndex</i>	評価するジョイントのインデックス
<i>evalTime</i>	評価時間（秒）

返 り 値

なし

解 説

この関数は、指定されたアニメーションおよび時刻において、特定のジョイントを評価します。

evalTime は強制的に（0、d）の範囲に補正されます。dはクリップの持続時間です。

注 意

この関数が用意されているのは、効率よりも利便性のためなので、用途として想定しているのは、例外的な場合や初期化目的だけです。一般的な評価の際には、SPU 処理関数を使ってください。

この関数は、ビットパックを有効にして作成されたアニメーションデータはサポートしません。

edgeAnimEvaluateUserChannel

指定されたアニメーションおよび時刻において特定のユーザチャンネルを評価する

定 義

```
#include <edge/anim/edgeanim_ppu.h>
float edgeAnimEvaluateUserChannel(
    const EdgeAnimAnimation* anim,
    const EdgeAnimSkeleton* skel,
    uint32_t channelIndex,
    float evalTime
)
```

呼 出 条 件

マルチスレッドセーフである。

引 数

<i>anim</i>	アニメーションへのポインタ
<i>skel</i>	スケルトンへのポインタ
<i>channelIndex</i>	評価するユーザチャンネルのインデックス
<i>evalTime</i>	評価時間（秒）

返 り 値

評価後のユーザチャンネルを返します

解 説

この関数は、特定のユーザチャンネルを、指定されたアニメーションおよび時刻において評価します。

evalTime は強制的に (0, d) の範囲に補正されます。ここで、*d* はクリップの持続時間です。

注 意

この関数が用意されているのは、効率よりも便利さのためなので、用途として想定しているのは、例外的な場合や初期化目的だけです。一般的な評価の際には、SPU 処理関数を使ってください。

この関数は、ビットパックを有効にして作成されたアニメーションデータはサポートしません。

SPU関数

edgeAnimSpuInitialize

SPU Edge Animation ライブラリを初期化する

定 義

```
#include <edge/anim/edgeanim_spu.h>
void edgeAnimSpuInitialize(
    EdgeAnimSpuContext* spuContext,
    const EdgeAnimPpuContext* ppuContext,
    uint32_t spuId,
    void* lsStorage,
    uint32_t sizeLsStorage,
    uint32_t numJoints,
    uint32_t numUserChannels,
    uint32_t maxSizeEvalBuffer,
    uint32_t maxSizeUserBuffer
)
```

引 数

<i>spuContext</i>	この関数によって書き込まれる SPU コンテキスト構造体のアドレス
<i>ppuContext</i>	PPU コンテキストへのポインタ
<i>spuId</i>	この SPU 固有の ID (0~5)
<i>lsStorage</i>	Edge Animation に使われるローカルストレージのアドレス
<i>sizeLsStorage</i>	Edge Animation が利用できるローカルストレージのサイズ
<i>numJoints</i>	全ジョイントの数
<i>numUserChannels</i>	全ユーザチャンネルの数
<i>maxSizeEvalBuffer</i>	評価バッファの最大サイズ (デフォルトでは EDGE_ANIM_EVAL_BUFFER_SIZE)
<i>maxSizeUserBuffer</i>	ユーザバッファの最大サイズ (デフォルトでは 0)

返 り 値

なし

解 説

この関数は、SPU Edge Animation ライブラリを初期化します。

lsStorage と *sizeLsStorage* には、Edge Animation で利用できるローカル記憶領域のアドレスおよびサイズを指定します。この関数は、バッファサイズが小さすぎる場合にはアサーションを実行します。必要な最小サイズは、以下のように計算されます。

$$3 * (\text{maxSizeEvalBuffer} + \text{maxSizeUserBuffer} + \text{sizePose})$$

これは、ローカルストアのポーズキャッシュに 3 つのスロットを割り当てます。追加で指定された領域は、すべて追加スロットに使われます。

maxSizeEvalBuffer は、評価バッファの最大サイズを指定します。このサイズは、任意のアニメーションに対してツールで指定した評価バッファのサイズ以上である必要があります。

maxSizeUserBuffer は、リーフやブランチのコールバックに渡されるユーザバッファのサイズを指定します。

edgeAnimSpuFinalize

SPU Edge Animation ライブラリの終了処理を行う

定 義

```
#include <edge/anim/edgeanim_spu.h>
void edgeAnimSpuFinalize(
    EdgeAnimSpuContext* spuContext
)
```

引 数

<i>spuContext</i>	SPU コンテキストへのポインタ
-------------------	------------------

返 り 値

なし

解 説

この関数は、SPU Edge Animation ライブラリの終了処理を行います。

edgeAnimProcessBlendTree

ブレンドツリーを処理して、その最終結果をポーズスタックの上に置く

定 義

```
#include <edge/anim/edgeanim_spu.h>
void edgeAnimProcessBlendTree(
    EdgeAnimSpuContext* spuContext,
    uint16_t rootIndex,
    const EdgeAnimBlendBranch* blendBranches,
    uint32_t numBranches,
    const EdgeAnimBlendLeaf* blendLeaves,
    uint32_t numLeaves,
    const EdgeAnimSkeleton* skeleton,
    const EdgeAnimMirrorPair* mirrorPairs,
    uint32_t numMirrorPairs,
    const EdgeAnimBranchCallback branchCallback,
    const EdgeAnimLeafCallback leafCallback
)
```

引 数

<i>spuContext</i>	SPU コンテキストへのポインタ
<i>rootIndex</i>	ブレンドツリーのルートインデックス
<i>blendBranches</i>	ブランチ配列へのポインタ
<i>numBranches</i>	ブランチの数
<i>blendLeaves</i>	リーフ配列へのポインタ
<i>numLeaves</i>	リーフの数
<i>skeleton</i>	スケルトンへのポインタ
<i>mirrorPairs</i>	ミラーリング指定を指すポインタ（デフォルトでは 0）
<i>numMirrorPairs</i>	ミラー指定に含まれるエントリ数（デフォルトでは 0）
<i>branchCallback</i>	ユーザブランチコールバック関数へのポインタ（デフォルトでは 0）
<i>leafCallback</i>	ユーザリーフコールバック関数へのポインタ（デフォルトでは 0）

rootIndex には、以下の 2 つのフラグのいずれかとインデックスとの間でビット OR をとった値を指定します。

マクロ	値	解説
EDGE_ANIM_BLEND_TREE_INDEX_LEAF	0x8000	リーフ指定子
EDGE_ANIM_BLEND_TREE_INDEX_BRANCH	0x4000	ブランチ指定子

返 り 値

なし

解 説

この関数は、ブレンドツリーを処理して、その最終結果をポーズスタックの上に置きます。

rootIndex には、ツリーのルートインデックス（通常は 0）を指定します。このインデックスでは、リーフかブランチかを明示的に示す必要があります。

ミラーリングが行われる場合には、この関数にミラーリング指定が渡されなければなりません。ミラーリングは、[EdgeAnimBlendLeaf](#)または[EdgeAnimBlendBranch](#)内部の適切なフラグを設定することによって、リーフまたはブランチ単位で有効化されます。

オプションでコールバックを指定することができます。コールバックの詳細については、[EdgeAnimBranchCallback\(\)](#)、および[EdgeAnimLeafCallback\(\)](#)を参照してください。

関 連 項 目

[EdgeAnimBlendBranch](#)、[EdgeAnimBlendLeaf](#)、[EdgeAnimBranchCallback](#)、[EdgeAnimLeafCallback](#)、[edgeAnimProcessBlendTree](#)、[edgeAnimProcessCommandList](#)

edgeAnimProcessCommandList

ブレンドツリーのコマンドリストを処理する

定 義

```
#include <edge/anim/edgeanim_spu.h>
void edgeAnimProcessCommandList(
    EdgeAnimSpuContext* spuContext,
    const EdgeAnimCommand* commandList,
    const EdgeAnimSkeleton* skeleton,
    const EdgeAnimMirrorPair* mirrorPairs,
    uint32_t numMirrorPairs,
    const EdgeAnimBranchCallback branchCallback,
    const EdgeAnimLeafCallback leafCallback,
    const EdgeAnimUserCallback userCallback
)
```

引 数

<i>spuContext</i>	SPU コンテキストへのポインタ
<i>commandList</i>	コマンドリストへのポインタ
<i>skeleton</i>	スケルトンへのポインタ
<i>mirrorPairs</i>	ミラーリング指定を指すポインタ (デフォルトでは 0)
<i>numMirrorPairs</i>	ミラー指定に含まれるエントリ数 (デフォルトでは 0)
<i>branchCallback</i>	ユーザブランチコールバック関数へのポインタ (デフォルトでは 0)
<i>leafCallback</i>	ユーザリーフコールバック関数へのポインタ (デフォルトでは 0)
<i>userCallback</i>	認識されないコマンド用のユーザコールバック関数へのポインタ (デフォルトでは 0)

返 り 値

なし

解 説

この関数は、ブレンドツリーのコマンドリストを処理します。この関数は、[edgeAnimProcessBlendTree\(\)](#) によって内部的に呼び出されます。

関 連 項 目

[EdgeAnimCommand](#)、[edgeAnimProcessBlendTree](#)

edgeAnimLocalJointsToWorldMatrices3x4

ローカル空間ジョイント変換の配列をワールド空間 3x4 行列の配列に変換する

定 義

```
#include <edge/anim/edgeanim_spu.h>
void edgeAnimLocalJointsToWorldMatrices3x4(
    void* outputMatrices,
    const EdgeAnimJointTransform* inputJoints,
    const EdgeAnimJointTransform* rootJoint,
    const int16_t* jointLinkage,
    uint32_t count
)
```

引 数

<i>outputMatrices</i>	出力ワールド行列配列へのポインタ
<i>inputJoints</i>	入力ローカルジョイント変換配列へのポインタ
<i>rootJoint</i>	ルートジョイント変換へのポインタ
<i>jointLinkage</i>	SIMD (Single Instruction, Multiple Data) 階層へのポインタ
<i>count</i>	SIMD 階層内のエントリ数

返 り 値

なし。

解 説

この関数は、ローカル空間ジョイント変換の配列をワールド空間 3x4 行列の配列に変換します。

[edgeAnimLocalJointsToWorldJoints](#)関数の場合と異なり、階層内の各ジョイントのローカルスケール要素が、最終的なジョイントのワールド姿勢ではなくその各ジョイント自身のワールド姿勢において、正しく変換されるかどうかのチェックが計算時に行われます。

SIMD の処理の特性上、*outputMatrices* 配列のサイズは 4 エントリの倍数でなければなりません。

*jointLinkage*には、[EdgeAnimSkeleton](#)に含まれる *simdHierarchy*配列のアドレスを渡します。

*count*は階層配列内のエントリ数を表しますが、これは、[EdgeAnimSkeleton](#)のデータから $4 * \text{numSimdHierarchyQuads}$ という式で計算できます。

この関数を使うと、オプションで親スケールを補うことができます。親ジョイントのインデックスに適切なビットが設定されている場合、親ジョイントのスケールを相殺するために、親ジョイントのローカル空間スケールの逆数に、子のローカル空間スケールが乗算されます。

注 意

SPU や SSE の実装では、あらゆるポインタが 16 バイトにアラインメントされている必要があります。

関 連 項 目

[EdgeAnimSkeleton](#)

edgeAnimLocalJointsToWorldMatrices4x4

ローカル空間ジョイント変換の配列をワールド空間 4x4 行列の配列に変換する

定 義

```
#include <edge/anim/edgeanim_spu.h>
void edgeAnimLocalJointsToWorldMatrices4x4 (
    void* outputMatrices,
    const EdgeAnimJointTransform* inputJoints,
    const EdgeAnimJointTransform* rootJoint,
    const int16_t* jointLinkage,
    uint32_t count
)
```

引 数

<i>outputMatrices</i>	出力ワールド行列配列へのポインタ
<i>inputJoints</i>	入力ローカルジョイント変換配列へのポインタ
<i>rootJoint</i>	ルートジョイント変換へのポインタ
<i>jointLinkage</i>	SIMD (Single Instruction, Multiple Data) 階層へのポインタ
<i>count</i>	SIMD 階層内のエントリ数

返 り 値

なし。

解 説

この関数は、ローカル空間ジョイント変換の配列をワールド空間 4x4 行列の配列に変換します。
[edgeAnimLocalJointsToWorldJoints\(\)](#) 関数の場合と異なり、階層内の各ジョイントのローカルスケール要素が、最終的なジョイントのワールド姿勢ではなくその各ジョイント自身のワールド姿勢において、正しく変換されるかどうかのチェックが計算時に行われます。

SIMD の処理の特性上、*outputMatrices* 配列のサイズは 4 エントリの倍数でなければなりません。
jointLinkage には、[EdgeAnimSkeleton](#) に含まれる *simdHierarchy* 配列のアドレスを渡します。
count は階層配列内のエントリ数を表しますが、これは、[EdgeAnimSkeleton](#) のデータから $4 * \text{numSimdHierarchyQuads}$ という式で計算できます。

親ジョイントのインデックスに適切なビットが設定されている場合、この関数は、オプションで親スケールを補うことができます。設定されている場合、親ジョイントのスケールを相殺するために、親ジョイントのローカル空間スケールの逆数に、子のローカル空間スケールが乗算されます。
 SPU や SSE の実装では、あらゆるポインタが 16 バイトにアラインメントされている必要があります。

関 連 項 目

[EdgeAnimSkeleton](#)

edgeAnimLocalJointsToWorldJoints

ローカル空間のジョイント変換配列をワールド空間に変換する

定 義

```
#include <edge/anim/edgeanim_spu.h>
void edgeAnimLocalJointsToWorldJoints (
    EdgeAnimJointTransform* outputJoints,
    const EdgeAnimJointTransform* inputJoints,
    const EdgeAnimJointTransform* rootJoint,
    const int16_t* jointLinkage,
    uint32_t count
)
```

引 数

<i>outputJoints</i>	出力先のワールドジョイント変換配列へのポインタ
<i>inputJoints</i>	入力元のローカルジョイント変換配列へのポインタ
<i>rootJoint</i>	ルートジョイント変換へのポインタ
<i>jointLinkage</i>	単一命令多重データ (SIMD) 階層へのポインタ
<i>count</i>	SIMD 階層のエントリの数

返 り 値

なし

解 説

この関数は、ローカル空間のジョイント変換配列をワールド空間に変換します。

SIMD の処理の特性上、*outputJoints* 配列のサイズは、4 エントリの倍数である必要があります。

jointLinkage は、[EdgeAnimSkeleton](#) 中の *simdHierarchy* 配列のアドレスを渡します。

count は、階層配列のエントリ数を指定します。この数は、[EdgeAnimSkeleton](#) のデータから $4 * \text{numSimdHierarchyQuads}$ という式により計算することができます。

親ジョイントのインデックスに適切なビットが設定されている場合、この関数は、オプションで親スケールを補うことができます。設定されている場合、親ジョイントのスケールリングを相殺するために、親ジョイントのローカル空間スケールの逆数に、子のローカル空間スケールが乗算されます。

注 意

SPU や SSE の実装では、あらゆるポインタが 16 バイトにアラインメントされている必要があります。

関 連 項 目

[EdgeAnimSkeleton](#)

edgeAnimWorldJointsToLocalJoints

ワールド空間のジョイント変換配列をローカル空間に変換する

定 義

```
#include <edge/anim/edgeanim_spu.h>
void edgeAnimWorldJointsToLocalJoints(
    EdgeAnimJointTransform* outputJoints,
    const EdgeAnimJointTransform* inputJoints,
    const EdgeAnimJointTransform* rootJoint,
    const int16_t* jointLinkage,
    uint32_t count
)
```

引 数

<i>outputJoints</i>	出力先のローカルジョイント変換配列へのポインタ
<i>inputJoints</i>	入力元のワールドジョイント変換配列へのポインタ
<i>rootJoint</i>	ルートジョイント変換へのポインタ
<i>jointLinkage</i>	単一命令多重データ (SIMD) 階層へのポインタ
<i>count</i>	SIMD 階層のエントリの数

返 り 値

なし

解 説

この関数は、ワールド空間のジョイント変換配列をローカル空間に変換します。

SIMD の処理の特性上、*outputJoints* 配列のサイズは、4 エントリの倍数である必要があります。

jointLinkage は、スケルトン中の *simdHierarchy* 配列のアドレスを渡します。

count は、階層配列のエントリ数を指定します。この数は、スケルトンのデータから $4 * \text{numSimdHierarchyQuads}$ という式により計算することができます。

edgeAnimJointsToMatrices4x4

ジョイント変換の配列を 4×4 行列に変換する

定 義

```
#include <edge/anim/edgeanim_spu.h>
void edgeAnimJointsToMatrices4x4(
    Vectormath::Aos::Transform3* outputMatrices,
    const EdgeAnimJointTransform* inputJoints,
    uint32_t count
)
```

引 数

<i>outputMatrices</i>	出力行列配列へのポインタ
<i>inputJoints</i>	入力ジョイント変換配列へのポインタ
<i>count</i>	ジョイント数

返 り 値

なし

解 説

この関数は、ジョイント変換の配列を 4×4 の行列に変換します。

SIMD の処理の特性上、*outputMatrices* 配列のサイズは、4 エントリの倍数である必要があります。

注 意

SPU や SSE の実装では、あらゆるポインタが 16 バイトにアラインメントされている必要があります。

edgeAnimMatrices4x4ToJoints

4×4 行列の配列をジョイント変換に変換する

定 義

```
#include <edge/anim/edgeanim_spu.h>
void edgeAnimMatrices4x4ToJoints(
    EdgeAnimJointTransform* outputJoints,
    const Vectormath::Aos::Transform3* inputMatrices,
    uint32_t count
)
```

引 数

<i>outputJoints</i>	出力先のジョイント変換配列へのポインタ
<i>inputMatrices</i>	入力元の行列配列へのポインタ
<i>count</i>	ジョイント数

返 り 値

なし

解 説

この関数は、4×4 行列の配列をジョイント変換に変換します。

SIMD の処理の特性上、*outputJoints* 配列のサイズは、4 エントリの倍数である必要があります。

注 意

SPU や SSE の実装では、あらゆるポインタが 16 バイトにアラインメントされている必要があります。

edgeAnimJointsToMatrices3x4

ジョイント変換の配列を 3×4 行列に変換する

定 義

```
#include <edge/anim/edgeanim_spu.h>
void edgeAnimJointsToMatrices3x4(
    void* outputMatrices,
    const EdgeAnimJointTransform* inputJoints,
    uint32_t count
)
```

引 数

<i>outputMatrices</i>	出力行列配列へのポインタ
<i>inputJoints</i>	入力ジョイント変換配列へのポインタ
<i>count</i>	ジョイント数

返 り 値

なし

解 説

この関数は、ジョイント変換の配列を 3×4 の行列に変換します。

SIMD の処理の特性上、*outputMatrices* 配列のサイズは、4 エントリの倍数である必要があります。

注 意

SPU や SSE の実装では、あらゆるポインタが 16 バイトにアラインメントされている必要があります。

edgeAnimMatrices3x4ToJoints

3×4 行列の配列をジョイント変換に変換する

定 義

```
#include <edge/anim/edgeanim_spu.h>
void edgeAnimMatrices3x4ToJoints(
    EdgeAnimJointTransform* outputJoints,
    const void* inputMatrices,
    uint32_t count
)
```

引 数

<i>outputJoints</i>	出力先のジョイント変換配列へのポインタ
<i>inputMatrices</i>	入力元の行列配列へのポインタ
<i>count</i>	ジョイント数

返 り 値

なし

解 説

この関数は、3×4 行列の配列をジョイント変換に変換します。

SIMD の処理の特性上、*outputJoints* 配列のサイズは、4 エントリの倍数である必要があります。

注 意

SPU や SSE の実装では、あらゆるポインタが 16 バイトにアラインメントされている必要があります。

edgeAnimBlendJointsLinear

2つのポーズの間の重み付きブレンドを実行する

定義

```
#include <edge/anim/edgeanim_spu.h>
void edgeAnimBlendJointsLinear(
    EdgeAnimJointTransform* outputJoints,
    uint8_t* outputWeights,
    const EdgeAnimJointTransform* leftJoints,
    const uint8_t* leftWeights,
    const EdgeAnimJointTransform* rightJoints,
    const uint8_t* rightWeights,
    float alpha,
    uint32_t count
)
```

引数

<i>outputJoints</i>	出力先のジョイント変換配列へのポインタ
<i>outputWeights</i>	出力先のジョイント重み配列へのポインタ
<i>leftJoints</i>	左ジョイント変換配列へのポインタ
<i>leftWeights</i>	左ジョイント重み配列へのポインタ
<i>rightJoints</i>	右ジョイント変換配列へのポインタ
<i>rightWeights</i>	右ジョイント重み配列へのポインタ
<i>alpha</i>	ブレンド係数 (0~1)
<i>count</i>	ジョイント数

返り値

なし

解説

この関数は、2つのポーズの間の重み付きブレンドを実行します。この関数は、[edgeAnimProcessCommandList\(\)](#) と [edgeAnimProcessBlendTree\(\)](#) によって内部的に呼び出されます。

- 回転は、球面線形補間を使ってブレンドされます。
- 平行移動とスケールは、線形補間を使ってブレンドされます。

以下の表に、（「PlayStation® Edge ライブラリ 概要」で説明した）部分的なアニメーションのロジックを要約しておきます。

定義されたジョイント	出力
なし	未定義
左	左
右	未定義
左および右	左と右のブレンド

注意

SIMD の処理の特性上、*outputJoints* 配列のサイズは、4 エントリの倍数である必要があります。

alpha は[0..1]の範囲にある必要があります。この範囲外の値に対する動作は未定義です。

leftWeights が NULL に設定された場合、左の重みはすべて 0xFF (1.0) であると仮定されます。

rightWeights が NULL に設定された場合、右の重みはすべて 0xFF (1.0) であると仮定されます。

この関数より高水準のインタフェースは、[edgeAnimBlendPose\(\)](#) で利用できます。

SPU や SSE 実装では、ジョイントポインタを 16 バイト、重みポインタを 4 バイトにアラインメントする必要があります。

関 連 項 目

[edgeAnimBlendPose](#)、[edgeAnimBlendUserLinear](#)

edgeAnimBlendJointsRelative

blendMode の値に応じて、2つのポーズの間の重みつき付加的・減算的なブレンド・合成を行う

定 義

```
#include <edge/anim/edgeanim_spu.h>
void edgeAnimBlendJointsRelative(
    EdgeAnimJointTransform* outputJoints,
    uint8_t* outputWeights,
    const EdgeAnimJointTransform* baseJoints,
    const uint8_t* baseWeights,
    const EdgeAnimJointTransform* deltaJoints,
    const uint8_t* deltaWeights,
    float alpha,
    EdgeAnimRelativeBlendMode blendMode,
    uint32_t count
)
```

引 数

<i>outputJoints</i>	出力先のジョイント変換配列へのポインタ
<i>outputWeights</i>	出力先のジョイント重み配列へのポインタ
<i>baseJoints</i>	ベースジョイント変換配列へのポインタ
<i>baseWeights</i>	ベースジョイント重み配列へのポインタ
<i>deltaJoints</i>	デルタジョイント変換配列へのポインタ
<i>deltaWeights</i>	デルタジョイント重み配列へのポインタ
<i>alpha</i>	ブレンド係数 (0~1)
<i>blendMode</i>	相対ブレンド/合成モード(下を参照)
<i>count</i>	ジョイント数

返 り 値

なし

解 説

この関数は、*blendMode*の値に応じて、2つのポーズの間の重みつき付加的・減算的なブレンド・合成を行うために使われます。この関数は、[edgeAnimProcessCommandList\(\)](#)と [edgeAnimProcessBlendTree\(\)](#)によって内部的に呼び出されます。

「デルタ」ブレンドモード：

```
blendMode = EDGE_ANIM_RELATIVE_BLEND_ADD_DELTA
```

これら3つのモードでは、この関数は左か、または左+右の2つのポーズの間の重み付きブレンドを実行します。

- 回転は、球面線形補間を使ってブレンドされます。
- 平行移動とスケールは、線形補間を使ってブレンドされます。

「合成」モード：

```
blendMode = EDGE_ANIM_RELATIVE_COMPOSE_ADD
blendMode = EDGE_ANIM_RELATIVE_COMPOSE_SUB
```

これら3つのモードでは、この関数は、ブレンディングを行わずに（アルファは無視）、左+右(ADD)もしくは左-右(SUB)の合成を行います。主な相違点は、左ジョイントが未定義の場合の動作です。

以下の表に、(*PlayStation®Edge* ライブラリ概要で説明した)部分的なアニメーションのロジックを要約します。

定義	BLEND_ADD_DELTA	COMPOSE_ADD	COMPOSE_SUB
なし	未定義	未定義	未定義
左	左	左	未定義
右	未定義	右	未定義
両方	左と、左+右との間のブレンド	左+右	左-右

注 意

SIMD の処理の特性上、*outputJoints* 配列のサイズは、4 エントリの倍数である必要があります。

alpha は[0..1]の範囲にある必要があります。この範囲外の値に対する動作は未定義です。

leftWeights が NULL に設定された場合、左の重みはすべて 0xFF (1.0) であると仮定されます。

rightWeights が NULL に設定された場合、右の重みはすべて 0xFF (1.0) であると仮定されます。

この関数より高水準のインタフェースは、[edgeAnimBlendPose\(\)](#) で利用できます。

SPU や SSE 実装では、ジョイントポインタを 16 バイト、重みポインタを 4 バイトにアラインメントする必要があります

関 連 項 目

[edgeAnimBlendPose](#)、[edgeAnimBlendUserRelative](#)

edgeAnimBlendPose

ポーズスタックのスロットの間のブレンド（ジョイントとユーザチャンネルの両方）を実行する

定 義

```
#include <edge/anim/edgeanim_spu.h>
void edgeAnimBlendPose(
    EdgeAnimSpuContext* spuContext,
    unsigned int poseDestDepth,
    unsigned int poseLeftDepth,
    unsigned int poseRightDepth,
    EdgeAnimBlendOp blendOp,
    float alpha,
    const EdgeAnimSkeleton* skeleton
)
```

引 数

<i>spuContext</i>	SPU コンテキストへのポインタ
<i>poseDestDepth</i>	出力先ポーズのスタック内での深度
<i>poseLeftDepth</i>	左ポーズのスタック内での深度
<i>poseRightDepth</i>	右ポーズのスタック内での深度
<i>blendOp</i>	ブレンド操作
<i>alpha</i>	ブレンド係数 (0~1)
<i>skeleton</i>	スケルトン定義へのポインタ

返 り 値

なし

解 説

この関数は、ポーズスタックのスロットの間のブレンド（ジョイントとユーザチャンネルの両方）を実行します。

注 意

この関数は、これらのスロットに影響を与えている DMA 操作が存在する場合には、DMA が完了するのを待ちます。

出力スロットは、入力と同じスロットでもかまいません。つまり、*poseDestDepth* に、*poseLeftDepth* もしくは *poseRightDepth* を指定すれば、その場で処理を行うことができます。

関 連 項 目

[edgeAnimBlendJointsLinear](#)、[edgeAnimBlendJointsRelative](#)、[edgeAnimBlendUserLinear](#)、[edgeAnimBlendUserRelative](#)

edgeAnimBlendUserLinear

2つのユーザチャンネル配列の間の線形ブレンディングを実行する

定 義

```
#include <edge/anim/edgeanim_spu.h>
void edgeAnimBlendUserLinear(
    float* outputChannels,
    uint8_t* outputWeights,
    const float* leftChannels,
    const uint8_t* leftWeights,
    const float* rightChannels,
    const uint8_t* rightWeights,
    const uint8_t* channelFlags,
    float alpha,
    uint32_t count
)
```

引 数

<i>outputChannels</i>	出力先のユーザチャンネル配列へのポインタ
<i>outputWeights</i>	出力先の重み配列へのポインタ
<i>leftChannels</i>	左ユーザチャンネル配列へのポインタ
<i>leftWeights</i>	左重み配列へのポインタ
<i>rightChannels</i>	右ユーザチャンネル配列へのポインタ
<i>rightWeights</i>	右重み配列へのポインタ
<i>channelFlags</i>	チャンネルフラグ配列へのポインタ
<i>alpha</i>	ブレンド係数 (0~1)
<i>count</i>	ユーザチャンネルの数

返 り 値

なし

解 説

この関数は、2つのユーザチャンネル配列の間の線形ブレンディングを実行します。この関数は、[edgeAnimProcessCommandList\(\)](#) と [edgeAnimProcessBlendTree\(\)](#) によって内部的に呼び出されます。

ブレンド関数の動作は、channelFlags で定義されたフラグによってチャンネル単位で変更することができます

- `EDGE_ANIM_USER_CHANNEL_FLAG_CLAMP01` が設定された場合には、出力値は `[0.0f...1.0f]` の範囲に強制的に補正されます。

それ以外のフラグは無視されます。

以下の表に、（「PlayStation®Edge ライブラリ概要」で説明した）部分的なアニメーションのロジックを要約します。

定義されたジョイント	出力
なし	未定義
左	左
右	未定義
左および右	左と右のブレンド

注 意

SIMD の処理の特性上、*outputChannels* 配列のサイズは、4 エントリの倍数である必要があります。

alpha は[0..1]の範囲にある必要があります。この範囲外の値に対する動作は未定義です。

leftWeights が NULL に設定された場合、左の重みはすべて 0xFF (1.0) であると仮定されます。

rightWeights が NULL に設定された場合、右の重みはすべて 0xFF (1.0) であると仮定されます。

channelFlags が NULL に設定された場合、フラグはすべて 0x00 (デフォルト動作) であると仮定されます。

この関数より高水準のインタフェースは、[edgeAnimBlendPose\(\)](#) で利用できます。

SPU や SSE 実装では、ジョイントポインタを 16 バイト、重みポインタを 4 バイトにアラインメントする必要があります。

関 連 項 目

[edgeAnimBlendPose](#)、[edgeAnimBlendJointsLinear](#)

edgeAnimBlendUserRelative

2つのユーザチャンネル配列の間の付加的ブレンディングを実行する

定 義

```
#include <edge/anim/edgeanim_spu.h>
void edgeAnimBlendUserRelative(
    float* outputChannels,
    uint8_t* outputWeights,
    const float* leftChannels,
    const uint8_t* leftWeights,
    const float* rightChannels,
    const uint8_t* rightWeights,
    const uint8_t* channelFlags,
    float alpha,
    EdgeAnimRelativeBlendMode blendMode,
    uint32_t count
)
```

引 数

<i>outputChannels</i>	出力先のユーザチャンネル配列へのポインタ
<i>outputWeights</i>	出力先の重み配列へのポインタ
<i>leftChannels</i>	ベースユーザチャンネル配列へのポインタ
<i>leftWeights</i>	ベース重み配列へのポインタ
<i>rightChannels</i>	デルタユーザチャンネル配列へのポインタ
<i>rightWeights</i>	デルタ重み配列へのポインタ
<i>channelFlags</i>	チャンネルフラグ配列へのポインタ
<i>alpha</i>	ブレンド係数 (0~1)
<i>blendMode</i>	相対ブレンド/合成モード
<i>count</i>	ユーザチャンネルの数

返 り 値

なし

解 説

この関数は、2つのユーザチャンネル配列の間の付加的ブレンディングを実行します。この関数は、[edgeAnimProcessCommandList\(\)](#)と[edgeAnimProcessBlendTree\(\)](#)によって内部的に呼び出されます。

ブレンド関数の動作は、*channelFlags*で定義されたフラグによってチャンネル単位で変更することができます

- `EDGE_ANIM_USER_CHANNEL_FLAG_CLAMP01` が設定された場合には、出力値は $[0.0f \dots 1.0f]$ の範囲に強制的に補正されます。
- `EDGE_ANIM_USER_CHANNEL_FLAG_MINMAX` が設定された場合には、「合成」モードの加算・減算操作は、すべて最大・最小操作に置換されます。

それ以外のフラグは無視されます。

「デルタ」ブレンドモード：

```
blendMode = EDGE_ANIM_RELATIVE_BLEND_ADD_DELTA
```

このモードでは、この関数は左、もしくは左+右のいずれかの、2つのポーズの間の重み付きブレンドを実行します。

- 回転は、球面線形補間を使ってブレンドされます。
- 平行移動とスケールは、線形補間を使ってブレンドされます。

「合成」モード：

```
blendMode = EDGE_ANIM_RELATIVE_COMPOSE_ADD
```

```
blendMode = EDGE_ANIM_RELATIVE_COMPOSE_SUB
```

このモードでは、この関数は、ブレンディングを行わずに（アルファは無視）、左 + 右（ADD）、もしくは左 - 右（SUB）のいずれかの合成を行います。主な相違点は、左ジョイントが未定義の場合の動作です。

以下の表に、（「PlayStation® Edge ライブラリ 概要」で説明した）部分的なアニメーションのロジックを要約します。

定義	BLEND_ADD_DELTA	COMPOSE_ADD	COMPOSE_SUB
なし	未定義	未定義	未定義
左	左	左	未定義
右	未定義	右	未定義
両方 (1)	ブレンド(左、 左+右)	左+右	左-右
両方 (2)	ブレンド(左、 左+右)	最大値(左、 右)	最小値(左、 右)

(1) 「デフォルト」モード—EDGE_ANIM_USER_CHANNEL_FLAG_MINMAX が設定されていない。

(2) 「ファジー」モード—EDGE_ANIM_USER_CHANNEL_FLAG_MINMAX が設定されている。

備 考

SIMD の処理の特性上、*outputChannels* 配列のサイズは、4 エントリの倍数である必要があります。

alpha は、[0..1]の範囲にある必要があります。この範囲外の値に対する動作は未定義です。

leftWeights が NULL に設定された場合、左の重みはすべて 0xFF (1.0) であると仮定されます。

rightWeights が NULL に設定された場合、右の重みはすべて 0xFF (1.0) であると仮定されます。

channelFlags が NULL に設定された場合、フラグはすべて 0x00（そのデフォルト動作）であると仮定されます。

この関数より高水準のインタフェースは、[edgeAnimBlendPose\(\)](#) で利用できます。

SPU や SSE 実装では、ジョイントポインタを 16 バイト、重みポインタを 4 バイトにアラインメントする必要があります。

関 連 項 目

[edgeAnimBlendPose](#)、[edgeAnimBlendJointsRelative](#)

edgeAnimPoseStackPush

スタックの一番上にポーズを 1 つ追加する

定 義

```
#include <edge/anim/edgeanim_spu.h>
void edgeAnimPoseStackPush(
    EdgeAnimSpuContext* spuContext
)
```

引 数

spuContext SPU コンテキストへのポインタ

返 り 値

なし

解 説

この関数は、スタックの一番上にポーズを 1 つ追加します。ローカルストアのスタック領域が一杯になった場合、この関数は、最後に使われたポーズの、この SPU に関連付けられたメインメモリの外部記憶領域（もしあれば）への DMA 転送を開始します。

ローカルストアにもメインメモリにも十分な領域が残っていない場合には、この関数はアサーションを発行します。

関 連 項 目

[edgeAnimPoseStackPop](#)

edgeAnimPoseStackPop

スタックの一番上にあるポーズを破棄する

定 義

```
#include <edge/anim/edgeanim_spu.h>
void edgeAnimPoseStackPop(
    EdgeAnimSpuContext* spuContext
)
```

引 数

<i>spuContext</i>	SPU コンテキストへのポインタ
-------------------	------------------

返 り 値

なし

解 説

この関数は、スタックの一番上にあるポーズを破棄します。メインメモリに保存されたポーズが存在する場合には、DMA を使って、最後に使われたポーズをローカルストアに書き込みます。

関 連 項 目

[edgeAnimPoseStackPush](#)

edgeAnimPoseStackGetPose

ポーズスタックの深さ *depth* の位置にあるポーズに結び付けられた情報を取得する

定 義

```
#include <edge/anim/edgeanim_spu.h>
void edgeAnimPoseStackGetPose(
    EdgeAnimSpuContext* spuContext,
    EdgeAnimPoseInfo* pose,
    uint32_t depth
)
```

引 数

<i>spuContext</i>	SPU コンテキストへのポインタ
<i>pose</i>	この関数によって書き込まれるポーズ情報構造体のアドレス
<i>depth</i>	ポーズのスタック深さ

返 り 値

なし

解 説

この関数は、ポーズスタックの深さ *depth* の位置にあるポーズに結び付けられた情報を取得します。

関 連 項 目

[EdgeAnimPoseInfo](#)

edgeAnimCustomDataChunk

指定されたカスタムデータテーブルから、エントリのハッシュ名で識別されるカスタムデータエントリを取得する

定 義

```
#include <edge/anim/edgeanim_spu.h>
Void* edgeAnimCustomDataChunk(
    const EdgeAnimCustomDataTable* pCustomDataTable,
    uint32_t chunkHash
)
```

引 数

<i>pCustomDataTable</i>	カスタムデータテーブルへのポインタ
<i>chunkHash</i>	要求されたカスタムデータエントリのハッシュ名

返 り 値

要求されたハッシュ名に対応するデータエントリへのポインタを返します。
ハッシュ名がテーブル内に存在しない場合は NULL を返します。

解 説

この関数は、指定されたカスタムデータテーブルから、エントリのハッシュ名で識別されるカスタムデータエントリを取得します。カスタムデータテーブルへのポインタは、[EdgeAnimSkeleton](#)・[EdgeAnimAnimation](#)構造体のオフセット値から決定できます。

関 連 項 目

[EdgeAnimCustomDataTable](#)

コールバック関数

EdgeAnimLeafCallback

リーフ処理の3つのパイプラインステージごとに呼び出される

定義

```
#include <edge/anim/edgeanim_spu.h>
void ( *EdgeAnimLeafCallback )(
    EdgeAnimSpuContext* spuContext,
    const EdgeAnimBlendLeaf const* leaf,
    const EdgeAnimAnimation* anim,
    const EdgeAnimSkeleton const* skel,
    const int32_t pipelineStage,
    const uint32_t dmaTag,
    void* userScratchBuffer
)
```

引数

<i>spuContext</i>	SPU コンテキストへのポインタ
<i>leaf</i>	ブレンドツリーのリーフへのポインタ
<i>anim</i>	アニメーションへのポインタ
<i>skel</i>	スケルトンへのポインタ
<i>pipelineStage</i>	現在のパイプラインステージ (-2、-1、0)
<i>dmaTag</i>	ユーザ DMA タグ
<i>userScratchBuffer</i>	ユーザスクラッチバッファ領域へのポインタ

返り値

なし

解説

このコールバック関数は、リーフ処理の3つのパイプラインステージごとに呼び出されます。このコールバックを登録するには、コールバックを [edgeAnimProcessBlendTree\(\)](#) の *leafCallback* 引数として渡します。

leaf には、現在処理中のリーフへのポインタが入っています。

anim には、パイプラインステージが 0 の場合にはアニメーションヘッダへのポインタ、それ以外の場合には NULL が入っています。

pipelineStage には、リーフの現在のパイプラインステージが入っています。この値として可能なのは、(発生順に) -2、-1、0 です。ステージ-2、-1 は、評価より前に発生するので、データのプリフェッチに利用できます。ステージ 0 では、リーフは処理済みであり、評価済みのポーズは、ポーズスタックの一番上のポーズとしてアクセスすることができます。

userScratchBuffer は、現在のリーフに結び付けられたユーザメモリを指しています。ユーザメモリがない場合には NULL が渡されます。ユーザスクラッチバッファのサイズは、[edgeAnimSpuInitialize\(\)](#) 呼び出しの中で指定します。

dmaTag は、ユーザスクラッチバッファとの間の DMA を開始するために使われます。前のパイプラインステージで開始されたユーザ DMA の結果を読む前にストールするのは、アプリケーションの責任です。

関 連 項 目

[edgeAnimProcessBlendTree](#)、[EdgeAnimBranchCallback](#)、[edgeAnimSpuInitialize](#)

EdgeAnimBranchCallback

ブランチ処理の3つのパイプラインステージごとに呼び出される

定義

```
#include <edge/anim/edgeanim_spu.h>
void ( *EdgeAnimBranchCallback) (
    EdgeAnimSpuContext* spuContext,
    const EdgeAnimBlendBranch const* branch,
    const EdgeAnimSkeleton const* skel,
    const int32_t pipelineStage,
    const uint32_t dmaTag,
    void* userScratchBuffer
)
```

引数

<i>spuContext</i>	SPU コンテキストへのポインタ
<i>branch</i>	ブレンドツリーのブランチへのポインタ
<i>skel</i>	スケルトンへのポインタ
<i>pipelineStage</i>	現在のパイプラインステージ (-2、-1、0)
<i>dmaTag</i>	ユーザ DMA タグ
<i>userScratchBuffer</i>	ユーザスクラッチバッファ領域へのポインタ

返り値

なし

解説

このコールバック関数は、ブランチ処理の3つのパイプラインステージごとに呼び出されます。このコールバックを登録するには、コールバックを [edgeAnimProcessBlendTree\(\)](#) の *branchCallback* 引数として渡します。

branch には、現在処理中のブランチへのポインタを指定します。

pipelineStage には、ブランチの現在のパイプラインステージを指定します。この値として可能なのは、(発生順に) -2、-1、0 です。ステージ-2、-1 は、ブレンディングより前に発生するので、データのプリフェッチに利用できます。ステージ0 では、ブランチは処理済みであり、ブレンド済みのポーズは、ポーズスタックの一番上のポーズとしてアクセスすることができます。

userScratchBuffer は、現在のブランチに結び付けられたユーザメモリを指しています。ユーザメモリがない場合にはNULLが渡されます。ユーザスクラッチバッファのサイズは、[edgeAnimSpuInitialize\(\)](#) 呼び出しの中で指定します。

dmaTag は、ユーザスクラッチバッファとの中の DMA を開始するために使われます。前のパイプラインステージで開始されたユーザ DMA の結果を読む前にストールするのは、アプリケーションの責任です。

関連項目

[edgeAnimProcessBlendTree](#)、[EdgeAnimLeafCallback](#)、[edgeAnimSpuInitialize](#)

EdgeAnimUserCallback

コマンド処理の3つのパイプラインステージごとに、認識されないコマンドIDが見つかるたびに呼び出される

定 義

```
#include <edge/anim/edgeanim_spu.h>
void ( *EdgeAnimUserCallback)(
    EdgeAnimSpuContext* spuContext,
    const EdgeAnimCommand const* cmd,
    const EdgeAnimSkeleton* skel,
    const int32_t pipelineStage,
    const uint32_t dmaTag,
    void* userScratchBuffer
)
```

引 数

<i>spuContext</i>	SPU コンテキストへのポインタ
<i>cmd</i>	認識されないユーザコマンドへのポインタ
<i>skel</i>	スケルトンへのポインタ
<i>pipelineStage</i>	現在のパイプラインステージ (-2、-1、0 のいずれか)
<i>dmaTag</i>	ユーザ DMA タグ
<i>userScratchBuffer</i>	ユーザスクラッチバッファ領域へのポインタ

返 り 値

なし。

解 説

このコールバック関数は、コマンド処理の3つのパイプラインステージごとに、認識されないコマンドIDが見つかるたびに呼び出されます。このコールバックを登録するには、これを [edgeAnimProcessCommandList\(\)](#) の *userCallback* 引数として渡します。

cmd には、認識されないユーザコマンドへのポインタが含まれます。

pipelineStage には、ブランチの現在のパイプラインステージが含まれます。値は-2、-1、0 のいずれかです（この順番で発生）。ステージ-2 と-1 はブレンド処理の前に発生し、データのプリフェッチ用として使用できます。ステージ0 ではブランチの処理がすでに実行されており、ブレンド後のポーズがポーズスタックの一番上のポーズとしてアクセス可能となります。

userScratchBuffer は、現在のコマンドに関連付けられたユーザメモリへのポインタです。ただし、何も指定されなかった場合はNULLになります。ユーザスクラッチバッファのサイズは、[edgeAnimSpuInitialize\(\)](#) の呼び出し時に指定されます。

dmaTag を使えば、ユーザスクラッチバッファに対する DMA 動作を開始できます。結果を読み取る前に、以前のパイプラインステージで開始されたすべてのユーザ DMA に対してストールを行うのは、アプリケーション側の責任になります。

関 連 項 目

[edgeAnimProcessCommandList](#)、[edgeAnimSpuInitialize](#)

定数

EdgeAnimBlendOp

さまざまなブレンディング演算モードを表す定数

定 義

マクロ	値	解説
EDGE_ANIM_BLENDOP_BLEND_LINEAR	0x000	以下のブレンド : 左[$\alpha = 0.0$] 右[$\alpha = 1.0$]
EDGE_ANIM_BLENDOP_BLEND_ADD_DELTA_RIGHT	0x001	以下のブレンド : 左[$\alpha = 0.0$] Add(左、右) [$\alpha = 1.0$]
EDGE_ANIM_BLENDOP_BLEND_ADD_DELTA_LEFT	0x010	以下のブレンド : 右[$\alpha = 0.0$] Add(右、左) [$\alpha = 1.0$]
EDGE_ANIM_BLENDOP_BLEND_SUB_DELTA_RIGHT	0x011	以下のブレンド : 左[$\alpha = 0.0$] Sub (左、右) [$\alpha = 1.0$]
EDGE_ANIM_BLENDOP_BLEND_SUB_DELTA_LEFT	0x100	以下のブレンド : 右[$\alpha = 0.0$] Sub (右、左) [$\alpha = 1.0$]
EDGE_ANIM_BLENDOP_COMPOSE_ADD	0x101	合成 : Add (左、右) [両方が指定された場合] それ以外の場合は、左もしくは右
EDGE_ANIM_BLENDOP_COMPOSE_SUB_RIGHT	0x110	合成 : Sub (左、右) [両方が指定された場合] それ以外の場合は、左もしくは Inverse(右)
EDGE_ANIM_BLENDOP_COMPOSE_SUB_LEFT	0x111	合成 : Sub (右、左) [両方が指定された場合] それ以外の場合は、右もしくは Inverse(左)

解 説

この定数は、[edgeAnimBlendPose\(\)](#) の *blendOp* 引数として渡されます。

EdgeAnimRelativeBlendMode

さまざまな相対ブレンド演算モードを表す定数

定 義

マクロ	値	解説
EDGE_ANIM_RELATIVE_BLEND_ADD_DELTA	0x000	「Base」と「Base +Delta」との間のブレンド
EDGE_ANIM_RELATIVE_BLEND_SUB_DELTA	0x001	「Base」と「Base - Delta」との間のブレンド
EDGE_ANIM_RELATIVE_COMPOSE_ADD	0x010	「Base + delta」の結果 ブレンドは行わない
EDGE_ANIM_RELATIVE_COMPOSE_SUB	0x011	「Base - delta」の結果 ブレンドは行わない
EDGE_ANIM_RELATIVE_NONE	0x100	ノーオペレーション（未定義）

解 説

この定数は、[edgeAnimBlendJointsRelative\(\)](#)および[edgeAnimBlendUserRelative\(\)](#)の`blendMode` 引数として渡されます。