Cutlib

2.0.0

作成： Doxygen 1.6.2

Wed Nov 10 13:29:54 2010

Contents

Chapter 1

モジュール索引

1.1 モジュール

すべてのモジュールの一覧です。

Chapter 2

構成索引

2.1 クラス階層

この継承一覧はおおまかにはソートされていますが、完全にアルファベット順でソートされてはいません。

Chapter 3

構成索引

3.1 構成

クラス、構造体、共用体、インタフェースの説明です。

Chapter 4

ファイル索引

4.1 ファイル一覧

これはファイル一覧です。

Chapter 5

モジュール

5.1 交点情報基本型

型定義

- typedef float cutlib::CutPos32 [6]

  交点座標基本型: **交点座標を** *float(32* **ビット***)* **として格納**

- typedef int32_t cutlib::CutPos8 [2]

  交点座標基本型: **交点座標を** *8* **ビット量子化して，** *3* **つずつ，** *2* **つの** *32* **ビット整数に格納**

- typedef int32_t cutlib::CutBid8 [2]

  境界 *ID* **基本型**: *0*〜*255* **の境界** *ID(8* **ビット***)* **を** *3* **つずつ，** *2* **つの** *32* **ビット整数に格納**

- typedef int32_t cutlib::CutBid5

  境界 *ID* **基本型**: *0*〜*31* **の境界** *ID(5* **ビット***)* **を** *6* **つまとめて，** *1* **つの** *32* **ビット整数に格納**

関数

- void cutlib::ClearCutPos (CutPos32 &cp)

  交点座標値を *1.0* **でクリア**

- void cutlib::SetCutPos (CutPos32 &cp, int d, float pos)

  交点座標値を設定 *(d* **方向***)*

- void cutlib::SetCutPos (CutPos32 &cp, const float pos[])

  交点座標値を設定 *(6* **方向まとめて***)*

- float cutlib::GetCutPos (const CutPos32 &cp, int d)
  交点座標値 *(d* **方向***)* **を得る**
- void cutlib::GetCutPos (const CutPos32 &cp, float pos[])
  交点座標値 *(6* **方向まとめて***)* **を得る**
- void cutlib::ClearCutPos (CutPos8 &cp)
  交点座標値を *1.0* **でクリア**
- void cutlib::SetCutPos (CutPos8 &cp, int d, float pos)
  交点座標値を設定 *(d* **方向***)*
- void cutlib::SetCutPos (CutPos8 &cp, const float pos[])
  交点座標値を設定 *(6* **方向まとめて***)*
- float cutlib::GetCutPos (const CutPos8 &cp, int d)
  交点座標値 *(d* **方向***)* **を得る**
- void cutlib::GetCutPos (const CutPos8 &cp, float pos[])
  交点座標値 *(6* **方向まとめて***)* **を得る**
- void cutlib::ClearCutBid (CutBid8 &cb)
  境界 *ID* **を** *0* **クリア**
- void cutlib::SetCutBid (CutBid8 &cb, int d, BidType bid)
  境界 *ID* **を設定** *(d* **方向***)*
- void cutlib::SetCutBid (CutBid8 &cb, const BidType bid[])
  境界 *ID* **を設定** *(6* **方向まとてて***)*
- BidType cutlib::GetCutBid (const CutBid8 &cb, int d)
  境界 *ID(d* **方向***)* **を得る**
- void cutlib::GetCutBid (const CutBid8 &cb, BidType bid[])
  境界 *ID(6* **方向まとめて***)* **を得る**
- template<typename CUT_POS >
  float cutlib::GetCutPos (const CUT_POS *cp, size_t ijk, int d)
  一次元配列データから交点座標を得る *(d* **方向***)*
- template<typename CUT_POS >
  void cutlib::GetCutPos (const CUT_POS *cp, size_t ijk, float pos[])
  一次元配列データから交点座標を得る *(6* **方向まとめて***)*
- template<typename CUT_BID >
  BidType cutlib::GetCutBid (const CUT_BID *cb, size_t ijk, int d)

一次元配列データから境界 *ID* を得る *(d* 方向*)*

- template<typename CUT_BID >
  void cutlib::GetCutBid (const CUT_BID *cb, size_t ijk, BidType bid[])
  一次元配列データから境界 *ID* を得る *(6* 方向まとめて*)*

変数

- const int32_t cutlib::CutPos8Clear0 [3]
  クリアマスク *(*方向別*)*
- const int32_t cutlib::CutPos8Clear = (int32_t)255 | ((int32_t)255<<8) | ((int32_t)255<<16)
  クリアマスク *(6* 方向まとめて*)*
- const int32_t cutlib::CutBid8Clear0 [3]
  クリアマスク *(*方向別*)*
- const int32_t cutlib::CutBid5Clear0 [6]
  クリアマスク *(*方向別*)*

5.1.1 関数

5.1.1.1 void cutlib::ClearCutBid (CutBid8 & *cb*) [inline]

境界 ID を 0 クリア

引数:

→ ***cb*** 境界 ID 基本型

5.1.1.2 void cutlib::ClearCutPos (CutPos8 & *cp*) [inline]

交点座標値を 1.0 でクリア

引数:

→ ***cp*** 交点座標基本型

5.1.1.3 void cutlib::ClearCutPos (CutPos32 & *cp*) [inline]

交点座標値を 1.0 でクリア

引数:

→ ***cp*** 交点座標基本型

5.1.1.4 template<typename CUT_BID > void cutlib::GetCutBid (const CUT_BID * *cb*, size_t *ijk*, BidType *bid*[ ]) [inline]

一次元配列データから境界 ID を得る (6 方向まとめて)

引数:

← ***cb*** 境界 ID 基本型配列

← ***ijk*** 一次元インデックス

→ ***bid*** 境界 ID 配列

5.1.1.5 template<typename CUT_BID > BidType cutlib::GetCutBid (const CUT_BID * *cb*, size_t *ijk*, int *d*) [inline]

一次元配列データから境界 ID を得る (d 方向)

引数:

← ***cb*** 境界 ID 基本型配列

← ***ijk*** 一次元インデックス

← ***d*** 交点探査方向 (0〜5)

戻り値:

境界 ID

5.1.1.6 void cutlib::GetCutBid (const CutBid8 & *cb*, BidType *bid*[ ]) [inline]

境界 ID(6 方向まとめて) を得る

引数:

← ***cb*** 境界 ID 基本型

→ ***bid*** 境界 ID 配列

5.1.1.7 BidType cutlib::GetCutBid (const CutBid8 & *cb*, int *d*) [inline]

境界 ID(d 方向) を得る

引数:

← ***cb*** 境界 ID 基本型
← ***d*** 交点探査方向 (0〜5)

戻り値:

境界 ID

5.1.1.8 template<typename CUT_POS > void cutlib::GetCutPos (const CUT_POS * *cp*, size_t *ijk*, float *pos*[ ]) [inline]

一次元配列データから交点座標を得る (6 方向まとめて)

引数:

← ***cp*** 交点座標基本型配列
← ***ijk*** 一次元インデックス
→ ***pos*** 交点座標値配列

5.1.1.9 template<typename CUT_POS > float cutlib::GetCutPos (const CUT_POS * *cp*, size_t *ijk*, int *d*) [inline]

一次元配列データから交点座標を得る (d 方向)

引数:

← ***cp*** 交点座標基本型配列
← ***ijk*** 一次元インデックス
← ***d*** 交点探査方向 (0〜5)

戻り値:

交点座標値

5.1.1.10 void cutlib::GetCutPos (const CutPos8 & *cp*, float *pos*[ ]) [inline]

交点座標値 (6 方向まとめて) を得る

引数:

← ***cp*** 交点座標基本型
→ ***pos*** 交点座標値配列

5.1.1.11 float cutlib::GetCutPos (const CutPos8 & *cp*, int *d*) [inline]

交点座標値 (d 方向) を得る

引数:

← ***cp*** 交点座標基本型
← ***d*** 交点探査方向 (0〜5)

戻り値:

交点座標値

5.1.1.12 void cutlib::GetCutPos (const CutPos32 & *cp*, float *pos*[ ]) [inline]

交点座標値 (6 方向まとめて) を得る

引数:

← ***cp*** 交点座標基本型
→ ***pos*** 交点座標値配列

5.1.1.13 float cutlib::GetCutPos (const CutPos32 & *cp*, int *d*) [inline]

交点座標値 (d 方向) を得る

引数:

← ***cp*** 交点座標基本型
← ***d*** 交点探査方向 (0〜5)

戻り値:

交点座標値

5.1.1.14 void cutlib::SetCutBid (CutBid8 & *cb*, const BidType *bid*[ ]) [inline]

境界 ID を設定 (6 方向まとてて)

引数:

→ ***cb*** 境界 ID 基本型
← ***bid*** 境界 ID 配列

5.1.1.15 void cutlib::SetCutBid (CutBid8 & *cb*, int *d*, BidType *bid*) [inline]

境界 ID を設定 (d 方向)

引数:

→ ***cb*** 境界 ID 基本型

← ***d*** 交点探査方向 (0〜5)

← ***bid*** 境界 ID

5.1.1.16 void cutlib::SetCutPos (CutPos8 & *cp*, const float *pos*[ ]) [inline]

交点座標値を設定 (6 方向まとめて)

引数:

→ ***cp*** 交点座標基本型

← ***pos*** 交点座標値配列

5.1.1.17 void cutlib::SetCutPos (CutPos8 & *cp*, int *d*, float *pos*) [inline]

交点座標値を設定 (d 方向)

引数:

→ ***cp*** 交点座標基本型

← ***d*** 交点探査方向 (0〜5)

← ***pos*** 交点座標値

5.1.1.18 void cutlib::SetCutPos (CutPos32 & *cp*, const float *pos*[ ]) [inline]

交点座標値を設定 (6 方向まとめて)

引数:

→ ***cp*** 交点座標基本型

← ***pos*** 交点座標値配列

5.1.1.19 void cutlib::SetCutPos (CutPos32 & *cp*, int *d*, float *pos*) [inline]

交点座標値を設定 (d 方向)

引数:

→ ***cp*** 交点座標基本型
← ***d*** 交点探査方向 (0〜5)
← ***pos*** 交点座標値

5.1.2 変数

5.1.2.1 const int32_t cutlib::CutBid5Clear0[6]

初期値:

```
 {
   ~(int32_t)31,
   ~((int32_t)31 << 5),
   ~((int32_t)31 << 10),
   ~((int32_t)31 << 15),
   ~((int32_t)31 << 20),
   ~((int32_t)31 << 25),
}
```

クリアマスク (方向別)

5.1.2.2 const int32_t cutlib::CutBid8Clear0[3]

初期値:

```
 {
   ~(int32_t)255,
   ~((int32_t)255 << 8),
   ~((int32_t)255 << 16)
}
```

クリアマスク (方向別)

5.1.2.3 const int32_t cutlib::CutPos8Clear0[3]

初期値:

```
 {
   ~(int32_t)255,
   ~((int32_t)255 << 8),
   ~((int32_t)255 << 16)
}
```

クリアマスク (方向別)

5.2 交点情報一次元配列ラッパクラス

構成

- class cutlib::CutInfoArray
  一次元配列ラッパクラスの基底クラス
- class cutlib::CutPosArray
  交点座標一次元配列ラッパ仮想クラス
- class cutlib::CutBidArray
  境界 *ID* **一次元配列ラッパ仮想クラス**
- class cutlib::CutPosArrayTemplate< CUT_POS >
  交点座標一次元配列ラッパクラステンプレート
- class cutlib::CutBidArrayTemplate< CUT_BID >
  境界 *ID* **一次元配列ラッパクラステンプレート**

型定義

- typedef CutPosArrayTemplate< CutPos32 > cutlib::CutPos32Array
  CutPos32 **型交点座標配列ラッパクラス**.
- typedef CutPosArrayTemplate< CutPos8 > cutlib::CutPos8Array
  CutPos8 **型交点座標配列ラッパクラス**.
- typedef CutBidArrayTemplate< CutBid8 > cutlib::CutBid8Array
  CutBid8 **型境界** *ID* **配列ラッパクラス**.
- typedef CutBidArrayTemplate< CutBid5 > cutlib::CutBid5Array
  CutBid5 **型境界** *ID* **配列ラッパクラス**.

5.3 交点情報 Octree セルデータアクセッサクラス

構成

- class cutlib::CutPosOctree
  Octree 交点座標データアクセッサ仮想クラス.
- class cutlib::CutBidOctree
  Octree 境界 *ID* データアクセッサ仮想クラス.
- class cutlib::CutPosOctreeTemplate< CUT_POS, SIZE_IN_FLOAT >
  Octree 交点座標データアクセサクラステンプレート.
- class cutlib::CutBidOctreeTemplate< CUT_BID, SIZE_IN_FLOAT >
  Octree 境界 *ID* データアクセサクラステンプレート.

型定義

- typedef CutPosOctreeTemplate< CutPos32, 6 > cutlib::CutPos32Octree
  CutPos32 型交点座標データアクセッサクラス.
- typedef CutPosOctreeTemplate< CutPos8, 2 > cutlib::CutPos8Octree
  CutPos8 型交点座標データアクセッサクラス.
- typedef CutBidOctreeTemplate< CutBid8, 2 > cutlib::CutBid8Octree
  CutBid8 型境界 *ID* データアクセッサクラス.
- typedef CutBidOctreeTemplate< CutBid5, 1 > cutlib::CutBid5Octree
  CutBid5 型境界 *ID* データアクセッサクラス.

5.4 交点情報計算関数

列挙型

- enum cutlib::CutlibReturn {
  cutlib::CL_SUCCESS = 0, cutlib::CL_BAD_GROUP_LIST = 1, cutlib::CL_BAD_POLYLIB = 2, cutlib::CL_BAD_SKLTREE = 3, cutlib::CL_SIZE_EXCEED = 4, cutlib::CL_OTHER_ERROR = 10 }
  交点情報計算関数リターンコード

関数

- CutlibReturn cutlib::CutInfoCell (const size_t ista[], const size_t nlen[], const float org[], const float d[], const Polylib *pl, CutPosArray *cutPos, CutBidArray *cutBid)
  交点情報計算: セル中心間, 配列ラッパクラス, 計算領域指定

- CutlibReturn cutlib::CutInfoCell (const float org[], const float d[], const Polylib *pl, CutPosArray *cutPos, CutBidArray *cutBid)
  交点情報計算: セル中心間, 配列ラッパクラス, 全領域

- CutlibReturn cutlib::CutInfoNode (const size_t ista[], const size_t nlen[], const float org[], const float d[], const Polylib *pl, CutPosArray *cutPos, CutBidArray *cutBid)
  交点情報計算: ノード間, 配列ラッパクラス, 計算領域指定

- CutlibReturn cutlib::CutInfoNode (const float org[], const float d[], const Polylib *pl, CutPosArray *cutPos, CutBidArray *cutBid)
  交点情報計算: ノード間, 配列ラッパクラス, 全領域

- template<typename CUT_POS , typename CUT_BID >
  CutlibReturn cutlib::CutInfoCell (const size_t ndim[], const size_t ista[], const size_t nlen[], const float org[], const float d[], const Polylib *pl, CUT_POS cutPos[], CUT_BID cutBid[])
  交点情報計算: セル中心間, 一次元配列, 計算領域指定

- template<typename CUT_POS , typename CUT_BID >
  CutlibReturn cutlib::CutInfoCell (const size_t ndim[], const float org[], const float d[], const Polylib *pl, CUT_POS cutPos[], CUT_BID cutBid[])
  交点情報計算: セル中心間, 一次元配列, 全領域

- template<typename CUT_POS , typename CUT_BID >
  CutlibReturn cutlib::CutInfoNode (const size_t ndim[], const size_t ista[], const size_t nlen[], const float org[], const float d[], const Polylib *pl, CUT_POS cutPos[], CUT_BID cutBid[])

交点情報計算: ノード間, 一次元配列, 計算領域指定

- template<typename CUT_POS , typename CUT_BID >
  CutlibReturn cutlib::CutInfoNode (const size_t ndim[], const float org[], const float d[], const Polylib *pl, CUT_POS cutPos[], CUT_BID cutBid[])

  交点情報計算: ノード間, 一次元配列, 全領域

5.4.1 列挙型

5.4.1.1 enum cutlib::CutlibReturn

交点情報計算関数リターンコード

列挙型の値:

CL_SUCCESS 成功

CL_BAD_GROUP_LIST (境界 ID, ポリゴングループ名) 対応リストが不正

CL_BAD_POLYLIB Polylib オブジェクトが不正 (未初期化等).

CL_BAD_SKLTREE SklTree オブジェクトが不正 (未初期化等).

CL_SIZE_EXCEED ista[]+nlen[] が配列サイズを越えている

CL_OTHER_ERROR その他のエラー

5.4.2 関数

5.4.2.1 template<typename CUT_POS , typename CUT_BID > CutlibReturn cutlib::CutInfoCell (const size_t *ndim*[], const float *org*[], const float *d*[], const Polylib * *pl*, CUT_POS *cutPos*[], CUT_BID *cutBid*[]) [inline]

交点情報計算: セル中心間, 一次元配列, 全領域

引数:

← ***ndim*** 全領域セル数

← ***org*** 領域原点座標

← ***d*** セル間隔

← ***pl*** Polylib クラスオブジェクト

→ ***cutPos*** 交点座標配列

→ ***cutBid*** 境界 ID 配列

5.4.2.2 template<typename CUT_POS , typename CUT_BID > CutlibReturn cutlib::CutInfoCell (const size_t *ndim*[ ], const size_t *ista*[ ], const size_t *nlen*[ ], const float *org*[ ], const float *d*[ ], const Polylib * *pl*, CUT_POS *cutPos*[ ], CUT_BID *cutBid*[ ]) [inline]

交点情報計算: セル中心間, 一次元配列, 計算領域指定

引数:

- ← ***ndim*** 全領域セル数
- ← ***ista*** 計算対象領域開始セル位置
- ← ***nlen*** 計算対象領域セル数
- ← ***org*** 領域原点座標
- ← ***d*** セル間隔
- ← ***pl*** Polylib クラスオブジェクト
- → ***cutPos*** 交点座標配列
- → ***cutBid*** 境界 ID 配列

5.4.2.3 CutlibReturn cutlib::CutInfoCell (const float *org*[ ], const float *d*[ ], const Polylib * *pl*, CutPosArray * *cutPos*, CutBidArray * *cutBid*)

交点情報計算: セル中心間, 配列ラッパクラス, 全領域

引数:

- ← ***org*** 領域原点座標
- ← ***d*** セル間隔
- ← ***pl*** Polylib クラスオブジェクト
- ↔ ***cutPos*** 交点座標配列ラッパ
- ↔ ***cutBid*** 境界 ID 配列ラッパ

5.4.2.4 CutlibReturn cutlib::CutInfoCell (const size_t *ista*[ ], const size_t *nlen*[ ], const float *org*[ ], const float *d*[ ], const Polylib * *pl*, CutPosArray * *cutPos*, CutBidArray * *cutBid*)

交点情報計算: セル中心間, 配列ラッパクラス, 計算領域指定

引数:

- ← ***ista*** 計算対象領域開始セル位置
- ← ***nlen*** 計算対象領域セル数

← ***org*** 領域原点座標

← ***d*** セル間隔

← ***pl*** Polylib クラスオブジェクト

↔ ***cutPos*** 交点座標配列ラッパ

↔ ***cutBid*** 境界 ID 配列ラッパ

5.4.2.5 template<typename CUT_POS , typename CUT_BID > CutlibReturn cutlib::CutInfoNode (const size_t *ndim*[ ], const float *org*[ ], const float *d*[ ], const Polylib * *pl*, CUT_POS *cutPos*[ ], CUT_BID *cutBid*[ ]) [inline]

交点情報計算: ノード間, 一次元配列, 全領域

引数:

← ***ndim*** 全領域セル数

← ***org*** 領域原点座標

← ***d*** セル間隔

← ***pl*** Polylib クラスオブジェクト

→ ***cutPos*** 交点座標配列

→ ***cutBid*** 境界 ID 配列

5.4.2.6 template<typename CUT_POS , typename CUT_BID > CutlibReturn cutlib::CutInfoNode (const size_t *ndim*[ ], const size_t *ista*[ ], const size_t *nlen*[ ], const float *org*[ ], const float *d*[ ], const Polylib * *pl*, CUT_POS *cutPos*[ ], CUT_BID *cutBid*[ ]) [inline]

交点情報計算: ノード間, 一次元配列, 計算領域指定

引数:

← ***ndim*** 全領域セル数

← ***ista*** 計算対象領域開始ノード位置

← ***nlen*** 計算対象領域ノード数

← ***org*** 領域原点座標

← ***d*** セル間隔

← ***pl*** Polylib クラスオブジェクト

→ ***cutPos*** 交点座標配列

→ ***cutBid*** 境界 ID 配列

5.4.2.7 CutlibReturn cutlib::CutInfoNode (const float *org*[ ], const float *d*[ ], const Polylib * *pl*, CutPosArray * *cutPos*, CutBidArray * *cutBid*)

交点情報計算: ノード間, 配列ラッパクラス, 全領域

引数:

← ***org*** 領域原点座標
← ***d*** セル間隔
← ***pl*** Polylib クラスオブジェクト
↔ ***cutPos*** 交点座標配列ラッパ
↔ ***cutBid*** 境界 ID 配列ラッパ

5.4.2.8 CutlibReturn cutlib::CutInfoNode (const size_t *ista*[ ], const size_t *nlen*[ ], const float *org*[ ], const float *d*[ ], const Polylib * *pl*, CutPosArray * *cutPos*, CutBidArray * *cutBid*)

交点情報計算: ノード間, 配列ラッパクラス, 計算領域指定

引数:

← ***ista*** 計算対象領域開始ノード位置
← ***nlen*** 計算対象領域ノード数
← ***org*** 領域原点座標
← ***d*** セル間隔
← ***pl*** Polylib クラスオブジェクト
↔ ***cutPos*** 交点座標配列ラッパ
↔ ***cutBid*** 境界 ID 配列ラッパ

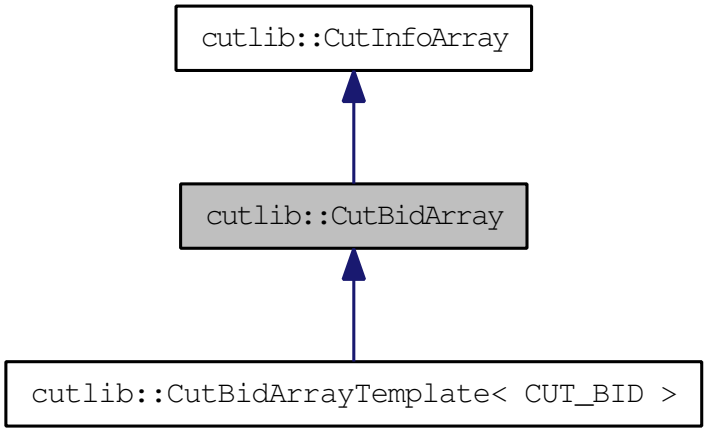
Chapter 6

クラス

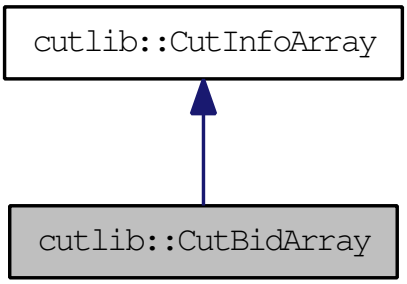
6.1 クラス cutlib::CutBidArray

境界 ID 一次元配列ラッパ仮想クラス

#include <CutInfoArray.h>cutlib::CutBidArray に対する継承グラフ



cutlib::CutBidArray のコラボレーション図



Public メソッド

- CutBidArray (size_t nx, size_t ny, size_t nz)
  コンストラクタ
- virtual ~CutBidArray ()
  デストラクタ
- virtual void setBid (size_t i, size_t j, size_t k, int d, BidType bid)=0
  境界 *ID* **を設定** *(d* **方向***)*
- virtual void setBid (size_t i, size_t j, size_t k, const BidType bid[])=0
  境界 *ID* **を設定** *(6* **方向まとてて***)*
- virtual BidType getBid (size_t i, size_t j, size_t k, int d) const =0
  境界 *ID(d* **方向***)* **を得る**
- virtual BidType getBid (size_t ijk, int d) const =0
  境界 *ID(d* **方向***)* **を得る** *(1* **次元インデックスで指定***)*
- virtual void getBid (size_t i, size_t j, size_t k, BidType bid[]) const =0
  境界 *ID(6* **方向まとめて***)* **を得る**
- virtual void getBid (size_t ijk, BidType bid[]) const =0
  境界 *ID(6* **方向まとめて***)* **を得る** *(1* **次元インデックスで指定***)*
- virtual void clear ()=0
  全配列データを *0* **クリア**

6.1.1 説明

境界 ID 一次元配列ラッパ仮想クラス

6.1.2 コンストラクタとデストラクタ

6.1.2.1 cutlib::CutBidArray::CutBidArray (size_t *nx*, size_t *ny*, size_t *nz*) [inline]

コンストラクタ

引数:

← ***nx,ny,nz*** 配列サイズ (3 次元で指定)

6.1.3 関数

6.1.3.1 virtual void cutlib::CutBidArray::getBid (size_t *ijk*, BidType *bid*[ ]) const [pure virtual]

境界 ID(6 方向まとめて) を得る (1 次元インデックスで指定)

引数:

← ***ijk*** 1 次元インデックス

→ ***bid*** 境界 ID 配列

cutlib::CutBidArrayTemplate< CUT_BID >で実装されています。

6.1.3.2 virtual void cutlib::CutBidArray::getBid (size_t *i*, size_t *j*, size_t *k*, BidType *bid*[ ]) const [pure virtual]

境界 ID(6 方向まとめて) を得る

引数:

← ***i,j,k*** 3 次元インデックス

→ ***bid*** 境界 ID 配列

cutlib::CutBidArrayTemplate< CUT_BID >で実装されています。

6.1.3.3 virtual BidType cutlib::CutBidArray::getBid (size_t *ijk*, int *d*) const [pure virtual]

境界 ID(d 方向) を得る (1 次元インデックスで指定)

引数:

← ***ijk*** 1 次元インデックス

← ***d*** 交点探査方向 (0〜5)

戻り値:

境界 ID

cutlib::CutBidArrayTemplate< CUT_BID >で実装されています。

6.1.3.4 virtual BidType cutlib::CutBidArray::getBid (size_t *i*, size_t *j*, size_t *k*, int *d*) const [pure virtual]

境界 ID(d 方向) を得る

引数:

← ***i,j,k*** 3 次元インデックス
← ***d*** 交点探査方向 (0〜5)

戻り値:

境界 ID

cutlib::CutBidArrayTemplate< CUT_BID >で実装されています。

6.1.3.5 virtual void cutlib::CutBidArray::setBid (size_t *i*, size_t *j*, size_t *k*, const BidType *bid*[ ]) [pure virtual]

境界 ID を設定 (6 方向まとてて)

引数:

← ***i,j,k*** 3 次元インデックス
← ***bid*** 境界 ID 配列

cutlib::CutBidArrayTemplate< CUT_BID >で実装されています。

6.1.3.6 virtual void cutlib::CutBidArray::setBid (size_t *i*, size_t *j*, size_t *k*, int *d*, BidType *bid*) [pure virtual]

境界 ID を設定 (d 方向)

引数:

← ***i,j,k*** 3 次元インデックス
← ***d*** 交点探査方向 (0〜5)
← ***bid*** 境界 ID

cutlib::CutBidArrayTemplate< CUT_BID >で実装されています。

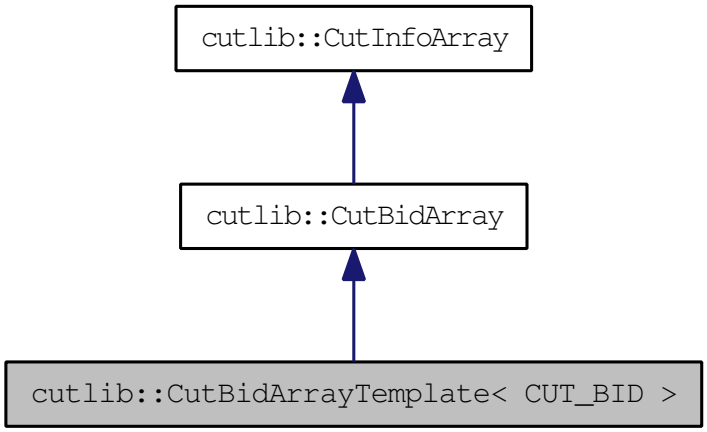
このクラスの説明は次のファイルから生成されました:

- include/CutInfo/CutInfoArray.h

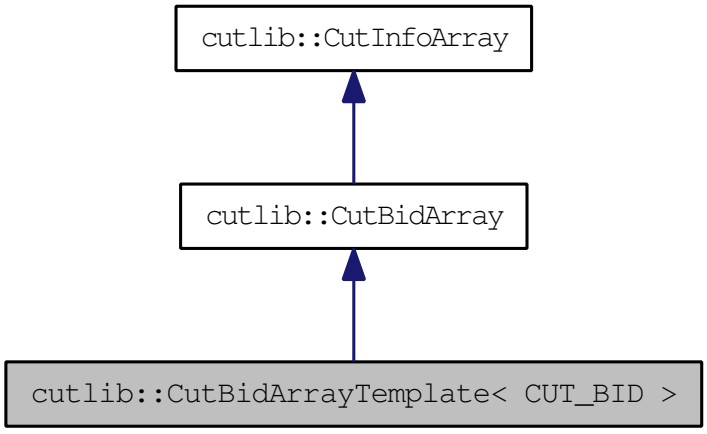
6.2 クラス テンプレート cutlib::CutBidArrayTemplate< CUT_BID >

境界 ID 一次元配列ラッパクラステンプレート

`#include <CutInfoArray.h>`cutlib::CutBidArrayTemplate< CUT_BID > に対する継承グラフ



cutlib::CutBidArrayTemplate< CUT_BID > のコラボレーション図



Public メソッド

- CutBidArrayTemplate (size_t nx, size_t ny, size_t nz)
  コンストラクタ *(自前で一次元データ領域を確保)*
- CutBidArrayTemplate (const size_t ndim[])

コンストラクタ *(自前で一次元データ領域を確保)*

- CutBidArrayTemplate (CUT_BID *data, size_t nx, size_t ny, size_t nz)
  コンストラクタ *(一次元データ領域をインポート)*
- ~CutBidArrayTemplate ()
  デストラクタ
- void setBid (size_t i, size_t j, size_t k, int d, BidType bid)
  境界 *ID* を設定 *(d* 方向*)*
- void setBid (size_t i, size_t j, size_t k, const BidType bid[])
  境界 *ID* を設定 *(6* 方向まとてて*)*
- BidType getBid (size_t i, size_t j, size_t k, int d) const
  境界 *ID(d* 方向*)* を得る
- BidType getBid (size_t ijk, int d) const
  境界 *ID(d* 方向*)* を得る *(1* 次元インデックスで指定*)*
- void getBid (size_t i, size_t j, size_t k, BidType bid[]) const
  境界 *ID(6* 方向まとめて*)* を得る
- void getBid (size_t ijk, BidType bid[]) const
  境界 *ID(6* 方向まとめて*)* を得る *(1* 次元インデックスで指定*)*
- CUT_BID * getDataPointer () const
  一次元配列データへのポインタを得る
- size_t getDataSize () const
  一次元配列データのサイズを得る
- void clear ()
  全配列データを *0* クリア

6.2.1 説明

template<typename CUT_BID> class cutlib::CutBidArrayTemplate< CUT_BID >

境界 ID 一次元配列ラッパクラステンプレート

6.2.2 コンストラクタとデストラクタ

6.2.2.1 template<typename CUT_BID > cutlib::CutBidArrayTemplate< CUT_BID >::CutBidArrayTemplate (size_t *nx*, size_t *ny*, size_t *nz*) [inline]

コンストラクタ (自前で一次元データ領域を確保) デストラクタで一次元データ領域を開放 (allocated_=true)

引数:

← ***nx,ny,nz*** 配列サイズ (3 次元で指定)

6.2.2.2 template<typename CUT_BID > cutlib::CutBidArrayTemplate< CUT_BID >::CutBidArrayTemplate (const size_t *ndim*[ ]) [inline]

コンストラクタ (自前で一次元データ領域を確保) デストラクタで一次元データ領域を開放 (allocated_=true)

引数:

← ***ndim*** 配列サイズ (3 次元で指定)

6.2.2.3 template<typename CUT_BID > cutlib::CutBidArrayTemplate< CUT_BID >::CutBidArrayTemplate (CUT_BID * *data*, size_t *nx*, size_t *ny*, size_t *nz*) [inline]

コンストラクタ (一次元データ領域をインポート) デストラクタで一次元データ領域を開放しない (allocated_=false)

引数:

← ***data*** 境界 ID 基本型配列

← ***nx,ny,nz*** 配列サイズ (3 次元で指定)

6.2.2.4 template<typename CUT_BID > cutlib::CutBidArrayTemplate< CUT_BID >::~CutBidArrayTemplate () [inline]

デストラクタ allocated_=true の場合のみ一次元データ領域を開放

6.2.3 関数

6.2.3.1 template<typename CUT_BID > void cutlib::CutBidArrayTemplate< CUT_BID >::getBid (size_t *ijk*, BidType *bid*[ ]) const [inline, virtual]

境界 ID(6 方向まとめて) を得る (1 次元インデックスで指定)

引数:

← ***ijk*** 1 次元インデックス

→ ***bid*** 境界 ID 配列

cutlib::CutBidArrayを実装しています。

6.2.3.2 template<typename CUT_BID > void cutlib::CutBidArrayTemplate< CUT_BID >::getBid (size_t *i*, size_t *j*, size_t *k*, BidType *bid*[ ]) const [inline, virtual]

境界 ID(6 方向まとめて) を得る

引数:

← ***i,j,k*** 3 次元インデックス

→ ***bid*** 境界 ID 配列

cutlib::CutBidArrayを実装しています。

6.2.3.3 template<typename CUT_BID > BidType cutlib::CutBidArrayTemplate< CUT_BID >::getBid (size_t *ijk*, int *d*) const [inline, virtual]

境界 ID(d 方向) を得る (1 次元インデックスで指定)

引数:

← ***ijk*** 1 次元インデックス

← ***d*** 交点探査方向 (0〜5)

戻り値:

境界 ID

cutlib::CutBidArrayを実装しています。

6.2.3.4 template<typename CUT_BID > BidType cutlib::CutBidArrayTemplate< CUT_BID >::getBid (size_t *i*, size_t *j*, size_t *k*, int *d*) const [inline, virtual]

境界 ID(d 方向) を得る

引数:

← ***i,j,k*** 3 次元インデックス
← ***d*** 交点探査方向 (0〜5)

戻り値:

境界 ID

cutlib::CutBidArrayを実装しています。

6.2.3.5 template<typename CUT_BID > void cutlib::CutBidArrayTemplate< CUT_BID >::setBid (size_t *i*, size_t *j*, size_t *k*, const BidType *bid*[]) [inline, virtual]

境界 ID を設定 (6 方向まとてて)

引数:

← ***i,j,k*** 3 次元インデックス
← ***bid*** 境界 ID 配列

cutlib::CutBidArrayを実装しています。

6.2.3.6 template<typename CUT_BID > void cutlib::CutBidArrayTemplate< CUT_BID >::setBid (size_t *i*, size_t *j*, size_t *k*, int *d*, BidType *bid*) [inline, virtual]

境界 ID を設定 (d 方向)

引数:

← ***i,j,k*** 3 次元インデックス
← ***d*** 交点探査方向 (0〜5)
← ***bid*** 境界 ID

cutlib::CutBidArrayを実装しています。

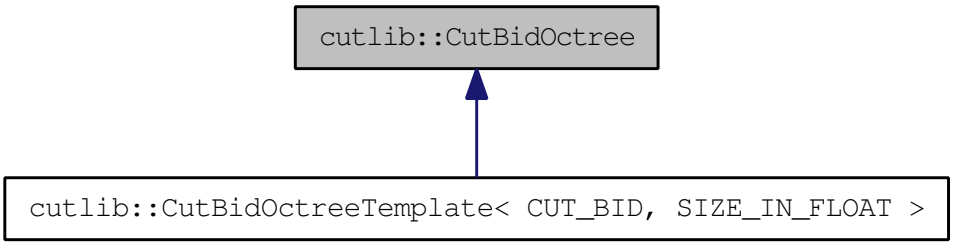
このクラスの説明は次のファイルから生成されました:

- include/CutInfo/CutInfoArray.h

6.3 クラス cutlib::CutBidOctree

Octree 境界 ID データアクセッサ仮想クラス.

`#include <CutInfoOctree.h>`cutlib::CutBidOctree に対する継承グラフ



Public メソッド

- CutBidOctree (int index)
  コンストラクタ
- virtual ~CutBidOctree ()
  デストラクタ
- virtual void assignData (float *data)=0
  SklCell **データ領域に結合**.
- virtual void setBid (int d, BidType bid)=0
  境界 *ID* **を設定** *(d* **方向***)*
- virtual void setBid (const BidType bid[])=0
  境界 *ID* **を設定** *(6* **方向まとてて***)*
- virtual BidType getBid (int d) const =0
  境界 *ID(d* **方向***)* **を得る**
- virtual void getBid (BidType bid[]) const =0
  境界 *ID(6* **方向まとめて***)* **を得る**
- virtual void clear ()=0
  6 **方向の境界** *ID* **を** *0* **クリア**
- virtual unsigned getSizeInFloat () const =0
  必要な格納領域サイズを *float* **単位で得る**

Protected 変数

- int index_

  SklCell データ領域内での格納開始インデックス.

6.3.1 説明

Octree 境界 ID データアクセッサ仮想クラス.

6.3.2 コンストラクタとデストラクタ

6.3.2.1 cutlib::CutBidOctree::CutBidOctree (int *index*) [inline]

コンストラクタ

引数:

← ***index*** SklCell データ領域内での格納開始インデックス

6.3.3 関数

6.3.3.1 virtual void cutlib::CutBidOctree::assignData (float * *data*) [pure virtual]

SklCell データ領域に結合.

引数:

← ***data*** SklCell データ領域ポインタ

cutlib::CutBidOctreeTemplate< CUT_BID, SIZE_IN_FLOAT >で実装されています。

6.3.3.2 virtual void cutlib::CutBidOctree::getBid (BidType *bid*[]) const [pure virtual]

境界 ID(6 方向まとめて) を得る

引数:

→ ***bid*** 境界 ID 配列

cutlib::CutBidOctreeTemplate< CUT_BID, SIZE_IN_FLOAT >で実装されています。

6.3.3.3 virtual BidType cutlib::CutBidOctree::getBid (int *d*) const [pure virtual]

境界 ID(d 方向) を得る

引数:

← ***d*** 交点探査方向 (0〜5)

戻り値:

境界 ID

cutlib::CutBidOctreeTemplate< CUT_BID, SIZE_IN_FLOAT >で実装されています。

6.3.3.4 virtual void cutlib::CutBidOctree::setBid (const BidType *bid*[ ]) [pure virtual]

境界 ID を設定 (6 方向まとてて)

引数:

← ***bid*** 境界 ID 配列

cutlib::CutBidOctreeTemplate< CUT_BID, SIZE_IN_FLOAT >で実装されています。

6.3.3.5 virtual void cutlib::CutBidOctree::setBid (int *d*, BidType *bid*) [pure virtual]

境界 ID を設定 (d 方向)

引数:

← ***d*** 交点探査方向 (0〜5)

← ***bid*** 境界 ID

cutlib::CutBidOctreeTemplate< CUT_BID, SIZE_IN_FLOAT >で実装されています。

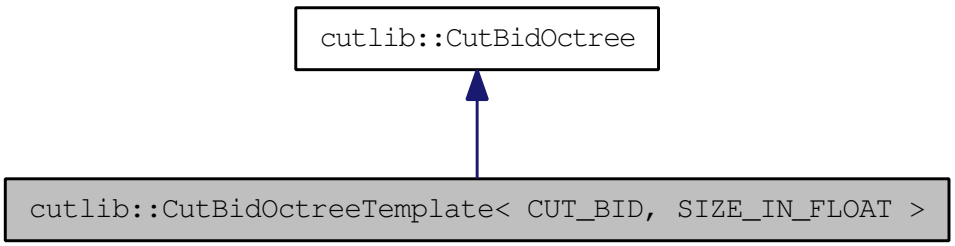
このクラスの説明は次のファイルから生成されました:

- include/CutInfo/CutInfoOctree.h

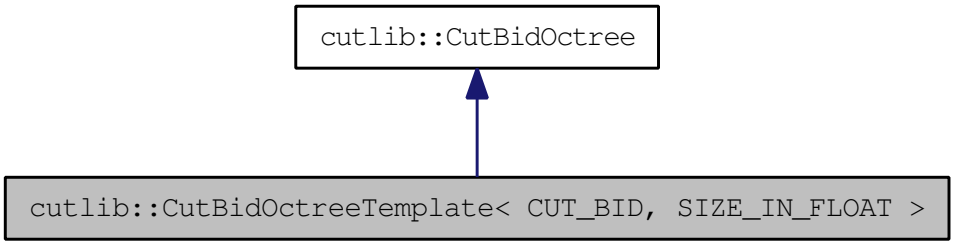
6.4 クラス テンプレート cutlib::CutBidOctreeTemplate< CUT_BID, SIZE_IN_FLOAT >

Octree 境界 ID データアクセサクラステンプレート.

`#include <CutInfoOctree.h>`cutlib::CutBidOctreeTemplate< CUT_BID, SIZE_IN_FLOAT > に対する継承グラフ



cutlib::CutBidOctreeTemplate< CUT_BID, SIZE_IN_FLOAT > のコラボレーション図



Public メソッド

- unsigned getSizeInFloat () const
  必要な格納領域サイズを *float* 単位で得る
- CutBidOctreeTemplate (int index)
  コンストラクタ
- virtual ~CutBidOctreeTemplate ()
  デストラクタ
- void assignData (float *data)
  ***SklCell* データ領域に結合**.
- void setBid (int d, BidType bid)
  境界 *ID* を設定 (*d* 方向)

- void setBid (const BidType bid[])

  境界 *ID* を設定 *(6* 方向まとてて*)*

- BidType getBid (int d) const

  境界 *ID(d* 方向*)* を得る

- void getBid (BidType bid[]) const

  境界 *ID(6* 方向まとめて*)* を得る

- void clear ()

  6 方向の境界 *ID* を *0* クリア

Static Public 変数

- static const unsigned SizeInFloat = SIZE_IN_FLOAT

  必要な格納領域サイズ *(float* 単位*)*

6.4.1 説明

template<typename CUT_BID, unsigned SIZE_IN_FLOAT> class cutlib::CutBidOctreeTemplate< CUT_BID, SIZE_IN_FLOAT >

Octree 境界 ID データアクセサクラステンプレート.

6.4.2 コンストラクタとデストラクタ

6.4.2.1 template<typename CUT_BID , unsigned SIZE_IN_FLOAT> cutlib::CutBidOctreeTemplate< CUT_BID, SIZE_IN_FLOAT >::CutBidOctreeTemplate (int *index*) [inline]

コンストラクタ

引数:

← ***index*** SklCell データ領域内での格納開始インデックス

6.4.3 関数

6.4.3.1 template<typename CUT_BID , unsigned SIZE_IN_FLOAT> void cutlib::CutBidOctreeTemplate< CUT_BID, SIZE_IN_FLOAT >::assignData (float * *data*) [inline, virtual]

SklCell データ領域に結合.

引数:

← ***data*** SklCell データ領域ポインタ

cutlib::CutBidOctreeを実装しています。

6.4.3.2 template<typename CUT_BID , unsigned SIZE_IN_FLOAT> void cutlib::CutBidOctreeTemplate< CUT_BID, SIZE_IN_FLOAT >::getBid (BidType *bid*[]) const [inline, virtual]

境界 ID(6 方向まとめて) を得る

引数:

→ ***bid*** 境界 ID 配列

cutlib::CutBidOctreeを実装しています。

6.4.3.3 template<typename CUT_BID , unsigned SIZE_IN_FLOAT> BidType cutlib::CutBidOctreeTemplate< CUT_BID, SIZE_IN_FLOAT >::getBid (int *d*) const [inline, virtual]

境界 ID(d 方向) を得る

引数:

← ***d*** 交点探査方向 (0〜5)

戻り値:

境界 ID

cutlib::CutBidOctreeを実装しています。

6.4.3.4 template<typename CUT_BID , unsigned SIZE_IN_FLOAT> void cutlib::CutBidOctreeTemplate< CUT_BID, SIZE_IN_FLOAT >::setBid (const BidType *bid*[]) [inline, virtual]

境界 ID を設定 (6 方向まとてて)

引数:

← ***bid*** 境界 ID 配列

cutlib::CutBidOctreeを実装しています。

6.4.3.5 template<typename CUT_BID , unsigned SIZE_IN_FLOAT> void cutlib::CutBidOctreeTemplate< CUT_BID, SIZE_IN_FLOAT >::setBid (int *d*, BidType *bid*) [inline, virtual]

境界 ID を設定 (d 方向)

引数:

← ***d*** 交点探査方向 (0〜5)

← ***bid*** 境界 ID

cutlib::CutBidOctreeを実装しています。

6.4.4 変数

6.4.4.1 template<typename CUT_BID , unsigned SIZE_IN_FLOAT> const unsigned cutlib::CutBidOctreeTemplate< CUT_BID, SIZE_IN_FLOAT >::SizeInFloat = SIZE_IN_FLOAT [static]

必要な格納領域サイズ (float 単位) テンプレートパラメータとして指定

このクラスの説明は次のファイルから生成されました:

- include/CutInfo/CutInfoOctree.h

6.5 クラス cutlib::CutBoundary

Polylib ポリゴングループ--境界 ID 対応付けクラス.

`#include <CutBoundary.h>`

Public メソッド

- CutBoundary (const std::string s, BidType i)
  コンストラクタ
- ~CutBoundary ()
  デストラクタ

Static Public メソッド

- static CutBoundaries * createCutBoundaryList (const Polylib *pl)
  Polylib **ポリゴングループ--境界** *ID* **対応リスト生成**.

Public 変数

- std::string name
  ポリゴングループ名
- BidType id
  境界 *ID*

6.5.1 説明

Polylib ポリゴングループ--境界 ID 対応付けクラス.

6.5.2 コンストラクタとデストラクタ

6.5.2.1 cutlib::CutBoundary::CutBoundary (const std::string *s*, BidType *i*) [inline]

コンストラクタ

引数:

s ポリゴングループ名

i 境界 ID

6.5.3 関数

6.5.3.1 CutBoundaries * cutlib::CutBoundary::createCutBoundaryList (const Polylib * *pl*) [static]

Polylib ポリゴングループ--境界 ID 対応リスト生成.

引数:

pl Polylib クラスオブジェクト

戻り値:

Polylib ポリゴングループ--境界 ID 対応リスト

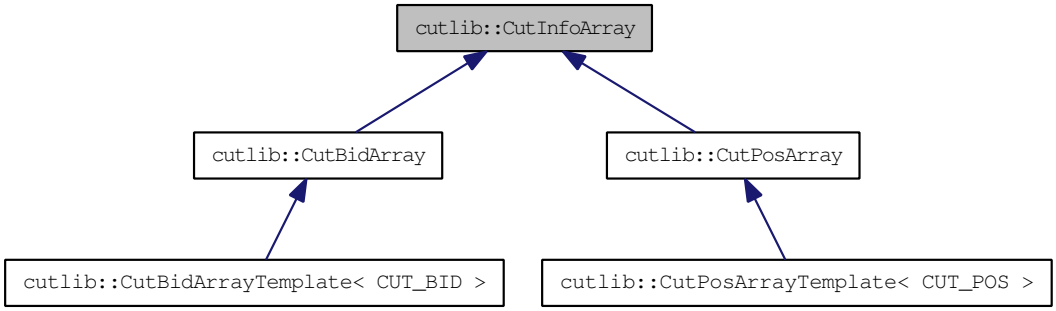
このクラスの説明は次のファイルから生成されました:

- src/CutBoundary.h
- src/CutBoundary.cpp

6.6 クラス cutlib::CutInfoArray

一次元配列ラッパクラスの基底クラス

`#include <CutInfoArray.h>`cutlib::CutInfoArray に対する継承グラフ



Public メソッド

- CutInfoArray (size_t nx, size_t ny, size_t nz)
  コンストラクタ
- virtual ~CutInfoArray ()
  デストラクタ
- size_t getSizeX () const
  x **方向のサイズを得る**
- size_t getSizeY () const
  y **方向のサイズを得る**
- size_t getSizeZ () const
  z **方向のサイズを得る**

Protected メソッド

- size_t index (size_t i, size_t j, size_t k) const
  3 **次元インデックス** *(i,j,k)* **より** *1* **次元インデックスを計算**

6.6.1 説明

一次元配列ラッパクラスの基底クラス

6.6.2 コンストラクタとデストラクタ

6.6.2.1 cutlib::CutInfoArray::CutInfoArray (size_t *nx*, size_t *ny*, size_t *nz*) [inline]

コンストラクタ

引数:

← ***nx,ny,nz*** 配列サイズ (3 次元で指定)

6.6.3 関数

6.6.3.1 size_t cutlib::CutInfoArray::index (size_t *i*, size_t *j*, size_t *k*) const [inline, protected]

3 次元インデックス (i,j,k) より 1 次元インデックスを計算

引数:

← ***i,j,k*** 3 次元インデックス

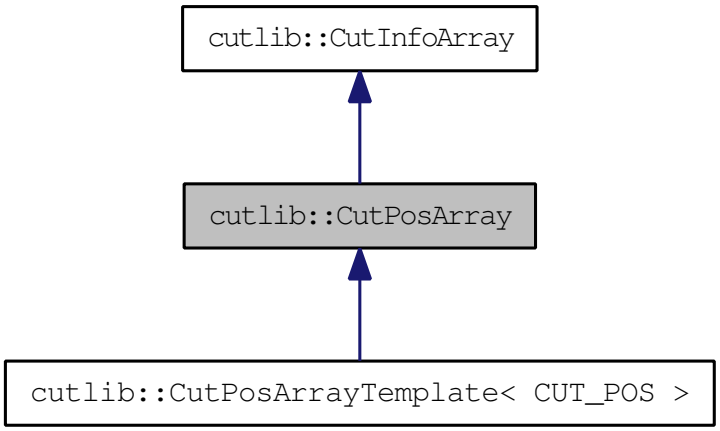
このクラスの説明は次のファイルから生成されました:

- include/CutInfo/CutInfoArray.h

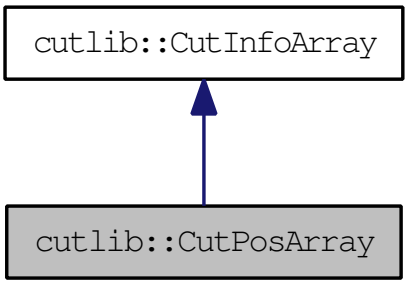
6.7 クラス cutlib::CutPosArray

交点座標一次元配列ラッパ仮想クラス

`#include <CutInfoArray.h>`cutlib::CutPosArray に対する継承グラフ



cutlib::CutPosArray のコラボレーション図



Public メソッド

- CutPosArray (size_t nx, size_t ny, size_t nz)
  コンストラクタ
- virtual ~CutPosArray ()
  デストラクタ
- virtual void setPos (size_t i, size_t j, size_t k, int d, float pos)=0
  交点座標値を設定 *(d* **方向***)*
- virtual void setPos (size_t i, size_t j, size_t k, const float pos[])=0

交点座標値を設定 *(6* 方向まとめて*)*

- virtual float getPos (size_t i, size_t j, size_t k, int d) const =0
  交点座標値 *(d* 方向*)* を得る
- virtual float getPos (size_t ijk, int d) const =0
  交点座標値 *(d* 方向*)* を得る *(1* 次元インデックスで指定*)*
- virtual void getPos (size_t i, size_t j, size_t k, float pos[ ]) const =0
  交点座標値 *(6* 方向まとめて*)* を得る
- virtual void getPos (size_t ijk, float pos[ ]) const =0
  交点座標値 *(6* 方向まとめて*)* を得る *(1* 次元インデックスで指定*)*
- virtual void clear ()=0
  全配列データを *1.0* でクリア

6.7.1 説明

交点座標一次元配列ラッパ仮想クラス

6.7.2 コンストラクタとデストラクタ

6.7.2.1 cutlib::CutPosArray::CutPosArray (size_t *nx*, size_t *ny*, size_t *nz*) [inline]

コンストラクタ

引数:

← ***nx,ny,nz*** 配列サイズ (3 次元で指定)

6.7.3 関数

6.7.3.1 virtual void cutlib::CutPosArray::getPos (size_t *ijk*, float *pos*[ ]) const [pure virtual]

交点座標値 (6 方向まとめて) を得る (1 次元インデックスで指定)

引数:

← ***ijk*** 1 次元インデックス

→ ***pos*** 交点座標配列

cutlib::CutPosArrayTemplate< CUT_POS >で実装されています。

6.7.3.2 virtual void cutlib::CutPosArray::getPos (size_t *i*, size_t *j*, size_t *k*, float *pos*[ ]) const [pure virtual]

交点座標値 (6 方向まとめて) を得る

引数:

← ***i,j,k*** 3 次元インデックス
→ ***pos*** 交点座標配列

cutlib::CutPosArrayTemplate< CUT_POS >で実装されています。

6.7.3.3 virtual float cutlib::CutPosArray::getPos (size_t *ijk*, int *d*) const [pure virtual]

交点座標値 (d 方向) を得る (1 次元インデックスで指定)

引数:

← ***ijk*** 1 次元インデックス
← ***d*** 交点探査方向 (0〜5)

戻り値:

交点座標値

cutlib::CutPosArrayTemplate< CUT_POS >で実装されています。

6.7.3.4 virtual float cutlib::CutPosArray::getPos (size_t *i*, size_t *j*, size_t *k*, int *d*) const [pure virtual]

交点座標値 (d 方向) を得る

引数:

← ***i,j,k*** 3 次元インデックス
← ***d*** 交点探査方向 (0〜5)

戻り値:

交点座標値

cutlib::CutPosArrayTemplate< CUT_POS >で実装されています。

6.7.3.5 virtual void cutlib::CutPosArray::setPos (size_t *i*, size_t *j*, size_t *k*, const float *pos*[ ]) [pure virtual]

交点座標値を設定 (6 方向まとめて)

引数:

← ***i,j,k*** 3 次元インデックス
← ***pos*** 交点座標配列

cutlib::CutPosArrayTemplate< CUT_POS >で実装されています。

6.7.3.6 virtual void cutlib::CutPosArray::setPos (size_t *i*, size_t *j*, size_t *k*, int *d*, float *pos*) [pure virtual]

交点座標値を設定 (d 方向)

引数:

← ***i,j,k*** 3 次元インデックス
← ***d*** 交点探査方向 (0〜5)
← ***pos*** 交点座標値

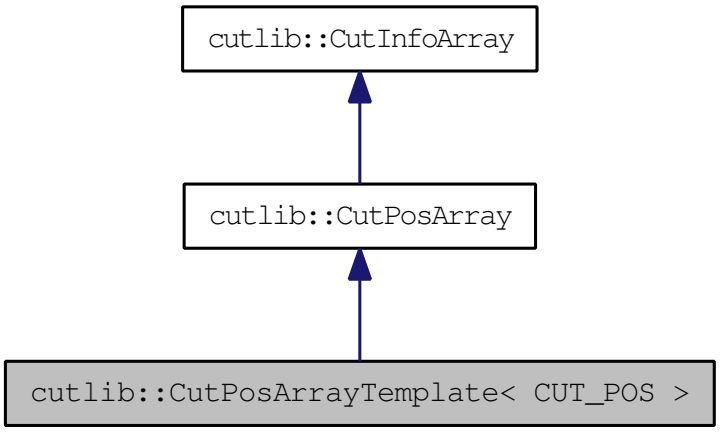
cutlib::CutPosArrayTemplate< CUT_POS >で実装されています。
このクラスの説明は次のファイルから生成されました:

- include/CutInfo/CutInfoArray.h

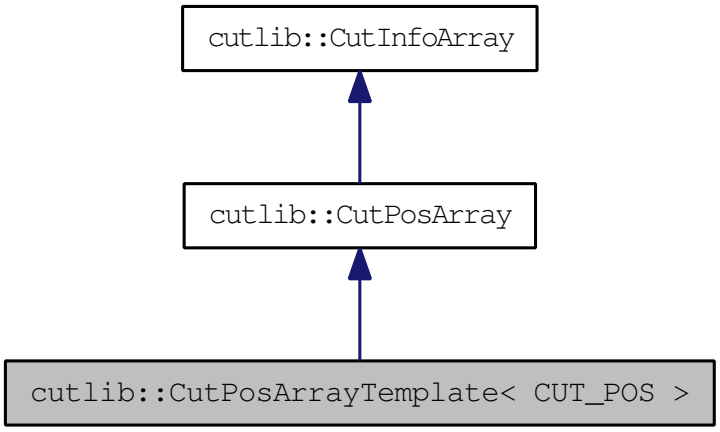
6.8 クラス テンプレート cutlib::CutPosArrayTemplate< CUT_POS >

交点座標一次元配列ラッパクラステンプレート

`#include <CutInfoArray.h>`cutlib::CutPosArrayTemplate< CUT_POS > に対する継承グラフ



cutlib::CutPosArrayTemplate< CUT_POS > のコラボレーション図



Public メソッド

- CutPosArrayTemplate (size_t nx, size_t ny, size_t nz)
  コンストラクタ *(自前で一次元データ領域を確保)*
- CutPosArrayTemplate (const size_t ndim[])

コンストラクタ *(自前で一次元データ領域を確保)*

- CutPosArrayTemplate (CUT_POS *data, size_t nx, size_t ny, size_t nz)

  コンストラクタ *(一次元データ領域をインポート)*

- ~CutPosArrayTemplate ()

  ディストラクタ

- void setPos (size_t i, size_t j, size_t k, int d, float pos)

  交点座標値を設定 *(d 方向)*

- void setPos (size_t i, size_t j, size_t k, const float pos[])

  交点座標値を設定 *(6 方向まとめて)*

- float getPos (size_t i, size_t j, size_t k, int d) const

  交点座標値 *(d 方向)* を得る

- float getPos (size_t ijk, int d) const

  交点座標値 *(d 方向)* を得る *(1 次元インデックスで指定)*

- void getPos (size_t i, size_t j, size_t k, float pos[]) const

  交点座標値 *(6 方向まとめて)* を得る

- void getPos (size_t ijk, float pos[]) const

  交点座標値 *(6 方向まとめて)* を得る *(1 次元インデックスで指定)*

- CUT_POS * getDataPointer () const

  一次元配列データへのポインタを得る

- size_t getDataSize () const

  一次元配列データのサイズを得る

- void clear ()

  全配列データを *1.0* でクリア

6.8.1 説明

template<typename CUT_POS> class cutlib::CutPosArrayTemplate< CUT_POS >

交点座標一次元配列ラッパクラステンプレート

6.8.2 コンストラクタとデストラクタ

6.8.2.1 template<typename CUT_POS > cutlib::CutPosArrayTemplate< CUT_POS >::CutPosArrayTemplate (size_t *nx*, size_t *ny*, size_t *nz*) [inline]

コンストラクタ (自前で一次元データ領域を確保) デストラクタで一次元データ領域を開放 (allocated_=true)

引数:

← ***nx,ny,nz*** 配列サイズ (3 次元で指定)

6.8.2.2 template<typename CUT_POS > cutlib::CutPosArrayTemplate< CUT_POS >::CutPosArrayTemplate (const size_t *ndim*[]) [inline]

コンストラクタ (自前で一次元データ領域を確保) デストラクタで一次元データ領域を開放 (allocated_=true)

引数:

← ***ndim*** 配列サイズ (3 次元で指定)

6.8.2.3 template<typename CUT_POS > cutlib::CutPosArrayTemplate< CUT_POS >::CutPosArrayTemplate (CUT_POS * *data*, size_t *nx*, size_t *ny*, size_t *nz*) [inline]

コンストラクタ (一次元データ領域をインポート) デストラクタで一次元データ領域を開放しない (allocated_=false)

引数:

← ***data*** 交点座標基本型配列

← ***nx,ny,nz*** 配列サイズ (3 次元で指定)

6.8.2.4 template<typename CUT_POS > cutlib::CutPosArrayTemplate< CUT_POS >::~CutPosArrayTemplate () [inline]

ディストラクタ allocated_=true の場合のみ一次元データ領域を開放

6.8.3 関数

6.8.3.1 template<typename CUT_POS > void cutlib::CutPosArrayTemplate< CUT_POS >::getPos (size_t *ijk*, float *pos*[ ]) const [inline, virtual]

交点座標値 (6 方向まとめて) を得る (1 次元インデックスで指定)

引数:

← ***ijk*** 1 次元インデックス

→ ***pos*** 交点座標配列

cutlib::CutPosArrayを実装しています。

6.8.3.2 template<typename CUT_POS > void cutlib::CutPosArrayTemplate< CUT_POS >::getPos (size_t *i*, size_t *j*, size_t *k*, float *pos*[ ]) const [inline, virtual]

交点座標値 (6 方向まとめて) を得る

引数:

← ***i,j,k*** 3 次元インデックス

→ ***pos*** 交点座標配列

cutlib::CutPosArrayを実装しています。

6.8.3.3 template<typename CUT_POS > float cutlib::CutPosArrayTemplate< CUT_POS >::getPos (size_t *ijk*, int *d*) const [inline, virtual]

交点座標値 (d 方向) を得る (1 次元インデックスで指定)

引数:

← ***ijk*** 1 次元インデックス

← ***d*** 交点探査方向 (0〜5)

戻り値:

交点座標値

cutlib::CutPosArrayを実装しています。

6.8.3.4 template<typename CUT_POS > float cutlib::CutPosArrayTemplate< CUT_POS >::getPos (size_t *i*, size_t *j*, size_t *k*, int *d*) const [inline, virtual]

交点座標値 (d 方向) を得る

引数:

← ***i,j,k*** 3 次元インデックス
← ***d*** 交点探査方向 (0〜5)

戻り値:

交点座標値

cutlib::CutPosArrayを実装しています。

6.8.3.5 template<typename CUT_POS > void cutlib::CutPosArrayTemplate< CUT_POS >::setPos (size_t *i*, size_t *j*, size_t *k*, const float *pos*[ ]) [inline, virtual]

交点座標値を設定 (6 方向まとめて)

引数:

← ***i,j,k*** 3 次元インデックス
← ***pos*** 交点座標配列

cutlib::CutPosArrayを実装しています。

6.8.3.6 template<typename CUT_POS > void cutlib::CutPosArrayTemplate< CUT_POS >::setPos (size_t *i*, size_t *j*, size_t *k*, int *d*, float *pos*) [inline, virtual]

交点座標値を設定 (d 方向)

引数:

← ***i,j,k*** 3 次元インデックス
← ***d*** 交点探査方向 (0〜5)
← ***pos*** 交点座標値

cutlib::CutPosArrayを実装しています。

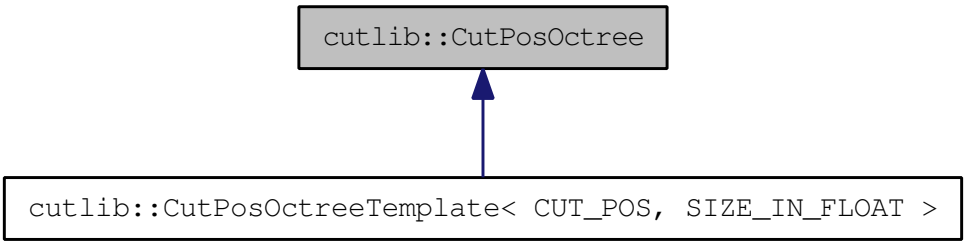
このクラスの説明は次のファイルから生成されました:

- include/CutInfo/CutInfoArray.h

6.9 クラス cutlib::CutPosOctree

Octree 交点座標データアクセッサ仮想クラス.

`#include <CutInfoOctree.h>`cutlib::CutPosOctree に対する継承グラフ



Public メソッド

- CutPosOctree (int index)
  コンストラクタ
- virtual ~CutPosOctree ()
  デストラクタ
- virtual void assignData (float *data)=0
  SklCell **データ領域に結合**.
- virtual void setPos (int d, float pos)=0
  交点座標値を設定 *(d* **方向***)*
- virtual void setPos (const float pos[])=0
  交点座標値を設定 *(6* **方向まとめて***)*
- virtual float getPos (int d) const =0
  交点座標値 *(d* **方向***)* **を得る**
- virtual void getPos (float pos[]) const =0
  交点座標値 *(6* **方向まとめて***)* **を得る**
- virtual void clear ()=0
  6 **方向の交点座標値を** *1.0* **でクリア**
- virtual unsigned getSizeInFloat () const =0
  必要な格納領域サイズを *float* **単位で得る**

Protected 変数

- int index_

  SklCell データ領域内での格納開始インデックス.

6.9.1 説明

Octree 交点座標データアクセッサ仮想クラス.

6.9.2 コンストラクタとデストラクタ

6.9.2.1 cutlib::CutPosOctree::CutPosOctree (int *index*) [inline]

コンストラクタ

引数:

← ***index*** SklCell データ領域内での格納開始インデックス

6.9.3 関数

6.9.3.1 virtual void cutlib::CutPosOctree::assignData (float * *data*) [pure virtual]

SklCell データ領域に結合.

引数:

← ***data*** SklCell データ領域ポインタ

cutlib::CutPosOctreeTemplate< CUT_POS, SIZE_IN_FLOAT >で実装されています。

6.9.3.2 virtual void cutlib::CutPosOctree::getPos (float *pos*[ ]) const [pure virtual]

交点座標値 (6 方向まとめて) を得る

引数:

→ ***pos*** 交点座標値配列

cutlib::CutPosOctreeTemplate< CUT_POS, SIZE_IN_FLOAT >で実装されています。

6.9.3.3 virtual float cutlib::CutPosOctree::getPos (int *d*) const [pure virtual]

交点座標値 (d 方向) を得る

引数:

← ***d*** 交点探査方向 (0〜5)

戻り値:

交点座標値

cutlib::CutPosOctreeTemplate< CUT_POS, SIZE_IN_FLOAT >で実装されています。

6.9.3.4 virtual void cutlib::CutPosOctree::setPos (const float *pos*[ ]) [pure virtual]

交点座標値を設定 (6 方向まとめて)

引数:

← ***pos*** 交点座標値配列

cutlib::CutPosOctreeTemplate< CUT_POS, SIZE_IN_FLOAT >で実装されています。

6.9.3.5 virtual void cutlib::CutPosOctree::setPos (int *d*, float *pos*) [pure virtual]

交点座標値を設定 (d 方向)

引数:

← ***d*** 交点探査方向 (0〜5)

← ***pos*** 交点座標値

cutlib::CutPosOctreeTemplate< CUT_POS, SIZE_IN_FLOAT >で実装されています。

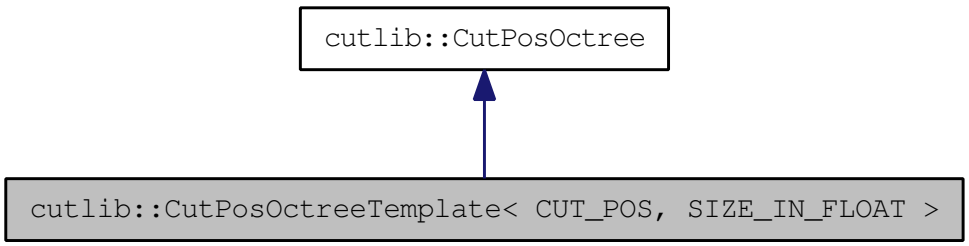
このクラスの説明は次のファイルから生成されました:

- include/CutInfo/CutInfoOctree.h

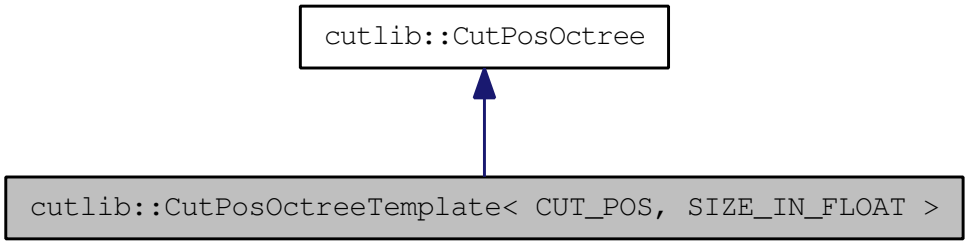
6.10 クラス テンプレート cutlib::CutPosOctreeTemplate< CUT_POS, SIZE_IN_FLOAT >

Octree 交点座標データアクセサクラステンプレート.

`#include <CutInfoOctree.h>`cutlib::CutPosOctreeTemplate< CUT_POS, SIZE_IN_FLOAT > に対する継承グラフ



cutlib::CutPosOctreeTemplate< CUT_POS, SIZE_IN_FLOAT > のコラボレーション図



Public メソッド

- unsigned getSizeInFloat () const
  必要な格納領域サイズを *float* 単位で得る
- CutPosOctreeTemplate (int index)
  コンストラクタ
- virtual ~CutPosOctreeTemplate ()
  デストラクタ
- void assignData (float *data)
  ***SklCell* データ領域に結合.**
- void setPos (int d, float pos)
  交点座標値を設定 (*d* 方向)

- void setPos (const float pos[])

  交点座標値を設定 *(6* **方向まとめて** *)*

- float getPos (int d) const

  交点座標値 *(d* **方向** *)* **を得る**

- void getPos (float pos[]) const

  交点座標値 *(6* **方向まとめて** *)* **を得る**

- void clear ()

  6 **方向の交点座標値を** *1.0* **でクリア**

Static Public 変数

- static const unsigned SizeInFloat = SIZE_IN_FLOAT

  必要な格納領域サイズ *(float* **単位** *)*

6.10.1 説明

template<typename CUT_POS, unsigned SIZE_IN_FLOAT> class cutlib::CutPosOctreeTemplate< CUT_POS, SIZE_IN_FLOAT >

Octree 交点座標データアクセサクラステンプレート.

6.10.2 コンストラクタとデストラクタ

6.10.2.1 template<typename CUT_POS , unsigned SIZE_IN_FLOAT> cutlib::CutPosOctreeTemplate< CUT_POS, SIZE_IN_FLOAT >::CutPosOctreeTemplate (int *index*) [inline]

コンストラクタ

引数:

← ***index*** SklCell データ領域内での格納開始インデックス

6.10.3 関数

6.10.3.1 template<typename CUT_POS , unsigned SIZE_IN_FLOAT> void cutlib::CutPosOctreeTemplate< CUT_POS, SIZE_IN_FLOAT >::assignData (float * *data*) [inline, virtual]

SklCell データ領域に結合.

引数:

← ***data*** SklCell データ領域ポインタ

cutlib::CutPosOctreeを実装しています。

6.10.3.2 template<typename CUT_POS , unsigned SIZE_IN_FLOAT> void cutlib::CutPosOctreeTemplate< CUT_POS, SIZE_IN_FLOAT >::getPos (float *pos*[]) const [inline, virtual]

交点座標値 (6 方向まとめて) を得る

引数:

→ ***pos*** 交点座標値配列

cutlib::CutPosOctreeを実装しています。

6.10.3.3 template<typename CUT_POS , unsigned SIZE_IN_FLOAT> float cutlib::CutPosOctreeTemplate< CUT_POS, SIZE_IN_FLOAT >::getPos (int *d*) const [inline, virtual]

交点座標値 (d 方向) を得る

引数:

← ***d*** 交点探査方向 (0〜5)

戻り値:

交点座標値

cutlib::CutPosOctreeを実装しています。

6.10.3.4 template<typename CUT_POS , unsigned SIZE_IN_FLOAT> void cutlib::CutPosOctreeTemplate< CUT_POS, SIZE_IN_FLOAT >::setPos (const float *pos*[]) [inline, virtual]

交点座標値を設定 (6 方向まとめて)

引数:

← ***pos*** 交点座標値配列

cutlib::CutPosOctreeを実装しています。

6.10.3.5 template<typename CUT_POS , unsigned SIZE_IN_FLOAT> void cutlib::CutPosOctreeTemplate< CUT_POS, SIZE_IN_FLOAT >::setPos (int *d*, float *pos*) [inline, virtual]

交点座標値を設定 (d 方向)

引数:

← ***d*** 交点探査方向 (0〜5)

← ***pos*** 交点座標値

cutlib::CutPosOctreeを実装しています。

6.10.4 変数

6.10.4.1 template<typename CUT_POS , unsigned SIZE_IN_FLOAT> const unsigned cutlib::CutPosOctreeTemplate< CUT_POS, SIZE_IN_FLOAT >::SizeInFloat = SIZE_IN_FLOAT [static]

必要な格納領域サイズ (float 単位) テンプレートパラメータとして指定

このクラスの説明は次のファイルから生成されました:

- include/CutInfo/CutInfoOctree.h

6.11 クラス cutlib::CutTriangle

BBox(binding box) 情報と境界 ID を持つカスタムポリゴンクラス.

`#include <CutTriangle.h>`

Public メソッド

- CutTriangle (Triangle *_t, BidType _bid)
  コンストラクタ
- bool intersectBox (const Vec3f &min, const Vec3f &max)
  三角形が直方体領域と交わるかの判定

Static Public メソッド

- static void AppendCutTriangles (CutTriangles &ctList, const Polylib *pl, const CutBoundaries *bList, const Vec3f &min, const Vec3f &max)
  Polylib **検索メソッドの結果をカスタムリストに追加**.
- static void CopyCutTriangles (const CutTriangles &ctListFrom, CutTriangles &ctListTo, const Vec3f &center, const Vec3f &d)
  直方体領域と交わる三角形のリストをコピー
- static void DeleteCutTriangles (CutTriangles &ctList)
  リスト内の三角形オブジェクトを消去

Public 変数

- Triangle * t
  Polylib **三角形ポリゴンクラス**.
- BidType bid
  境界 *ID*
- Vec3f bboxMin
  BBox **最小値**.
- Vec3f bboxMax
  BBox **最大値**.

6.11.1 説明

BBox(binding box) 情報と境界 ID を持つカスタムポリゴンクラス.

6.11.2 コンストラクタとデストラクタ

6.11.2.1 cutlib::CutTriangle::CutTriangle (Triangle * _t, BidType _bid)

コンストラクタ

引数:

← ***_t*** Polylib 三角形ポリゴンクラス

← ***_bid*** 境界 ID

6.11.3 関数

6.11.3.1 void cutlib::CutTriangle::AppendCutTriangles (CutTriangles & *ctList*, const Polylib * *pl*, const CutBoundaries * *bList*, const Vec3f & *min*, const Vec3f & *max*) [static]

Polylib 検索メソッドの結果をカスタムリストに追加.

引数:

↔ ***ctList*** 三角形リスト

← ***pl*** Polylib クラスオブジェクト

← ***bList*** (境界 ID, ポリゴングループ名) 対応リスト

← ***min,max*** 検索領域

6.11.3.2 void cutlib::CutTriangle::CopyCutTriangles (const CutTriangles & *ctListFrom*, CutTriangles & *ctListTo*, const Vec3f & *center*, const Vec3f & *d*) [static]

直方体領域と交わる三角形のリストをコピー

引数:

← ***ctListFrom*** 三角形リスト コピー元

→ ***ctListTo*** 三角形リスト コピー先

← ***center*** 直方体領域中心

← ***d*** 直方体領域幅の 1/2

6.11.3.3 void cutlib::CutTriangle::DeleteCutTriangles (CutTriangles & *ctList*) [static]

リスト内の三角形オブジェクトを消去

引数:

↔ ***ctList*** 三角形リスト

6.11.3.4 bool cutlib::CutTriangle::intersectBox (const Vec3f & *min*, const Vec3f & *max*)

三角形が直方体領域と交わるかの判定

引数:

← ***min*,*max*** 直方体頂点座標

戻り値:

true:交わる/false:交わらない

このクラスの説明は次のファイルから生成されました:

- src/CutTriangle.h
- src/CutTriangle.cpp

6.12 クラス cutlib::Timing

時間測定ストップウオッチクラス

`#include <CutTiming.h>`

Static Public メソッド

- static void Start (const string &name)
  ストップウオッチ *name* をスタート
- static void Stop (const string &name)
  ストップウオッチ *name* をストップ
- static void Print (const string &name)
  ストップウオッチ *name* の内容を表示

フレンド

- ostream & **operator**<< (ostream &os, const Timing &t)

6.12.1 説明

時間測定ストップウオッチクラス

6.12.2 関数

6.12.2.1 void cutlib::Timing::Print (const string & *name*) [static]

ストップウオッチ name の内容を表示

引数:

← ***name*** ストップウオッチ名

6.12.2.2 void cutlib::Timing::Start (const string & *name*) [static]

ストップウオッチ name をスタート name が未登録の場合は、新に作成

引数:

← ***name*** ストップウオッチ名

6.12.2.3 void cutlib::Timing::Stop (const string & *name*) [static]

ストップウオッチ name をストップ

引数:

← ***name*** ストップウオッチ名

このクラスの説明は次のファイルから生成されました:

- src/CutTiming.h
- src/CutTiming.cpp

Chapter 7

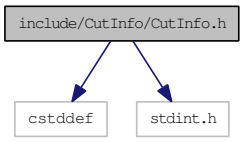
ファイル

7.1 include/CutInfo/CutInfo.h

交点情報基本データ型`#include <cstddef>`

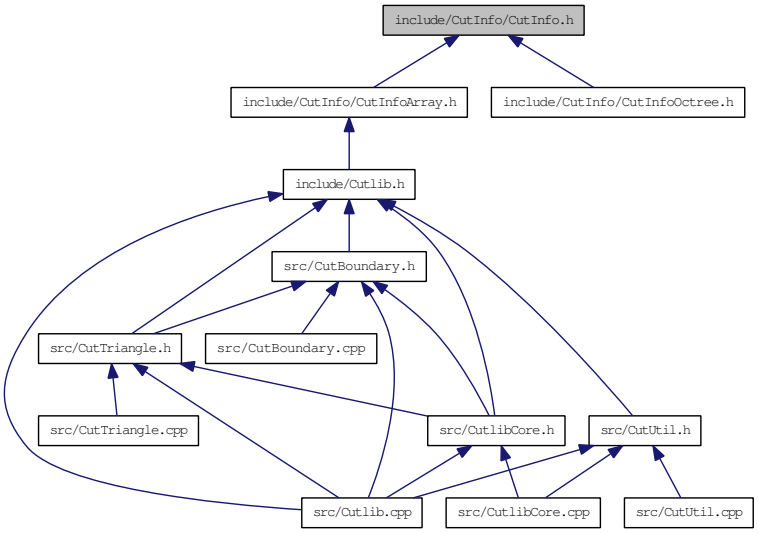
`#include <stdint.h>`

CutInfo.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示

しています。



型定義

- typedef unsigned char cutlib::BidType
  境界 *ID* 型

- typedef float cutlib::CutPos32 [6]
  交点座標基本型: 交点座標を *float(32* ビット*)* として格納

- typedef int32_t cutlib::CutPos8 [2]
  交点座標基本型: 交点座標を *8* ビット量子化して，*3* つずつ，*2* つの *32* ビット整数に格納

- typedef int32_t cutlib::CutBid8 [2]
  境界 *ID* 基本型: *0*〜*255* の境界 *ID(8* ビット*)* を *3* つずつ，*2* つの *32* ビット整数に格納

- typedef int32_t cutlib::CutBid5
  境界 *ID* 基本型: *0*〜*31* の境界 *ID(5* ビット*)* を *6* つまとめて，*1* つの *32* ビット整数に格納

列挙型

- enum **CutInfoOrder** {
  X_M, **X_P**, **Y_M**, **Y_P**,
  Z_M, **Z_P** }

交点探査方向の順番

関数

- void cutlib::ClearCutPos (CutPos32 &cp)
  交点座標値を *1.0* でクリア
- void cutlib::SetCutPos (CutPos32 &cp, int d, float pos)
  交点座標値を設定 *(d* 方向*)*
- void cutlib::SetCutPos (CutPos32 &cp, const float pos[])
  交点座標値を設定 *(6* 方向まとめて*)*
- float cutlib::GetCutPos (const CutPos32 &cp, int d)
  交点座標値 *(d* 方向*)* を得る
- void cutlib::GetCutPos (const CutPos32 &cp, float pos[])
  交点座標値 *(6* 方向まとめて*)* を得る
- void cutlib::ClearCutPos (CutPos8 &cp)
  交点座標値を *1.0* でクリア
- void cutlib::SetCutPos (CutPos8 &cp, int d, float pos)
  交点座標値を設定 *(d* 方向*)*
- void cutlib::SetCutPos (CutPos8 &cp, const float pos[])
  交点座標値を設定 *(6* 方向まとめて*)*
- float cutlib::GetCutPos (const CutPos8 &cp, int d)
  交点座標値 *(d* 方向*)* を得る
- void cutlib::GetCutPos (const CutPos8 &cp, float pos[])
  交点座標値 *(6* 方向まとめて*)* を得る
- void cutlib::ClearCutBid (CutBid8 &cb)
  境界 *ID* を *0* クリア
- void cutlib::SetCutBid (CutBid8 &cb, int d, BidType bid)
  境界 *ID* を設定 *(d* 方向*)*
- void cutlib::SetCutBid (CutBid8 &cb, const BidType bid[])
  境界 *ID* を設定 *(6* 方向まとてて*)*
- BidType cutlib::GetCutBid (const CutBid8 &cb, int d)
  境界 *ID(d* 方向*)* を得る

- void cutlib::GetCutBid (const CutBid8 &cb, BidType bid[])
  境界 *ID(6* 方向まとめて*)* を得る

- template<typename CUT_POS >
  float cutlib::GetCutPos (const CUT_POS *cp, size_t ijk, int d)
  一次元配列データから交点座標を得る *(d* 方向*)*

- template<typename CUT_POS >
  void cutlib::GetCutPos (const CUT_POS *cp, size_t ijk, float pos[])
  一次元配列データから交点座標を得る *(6* 方向まとめて*)*

- template<typename CUT_BID >
  BidType cutlib::GetCutBid (const CUT_BID *cb, size_t ijk, int d)
  一次元配列データから境界 *ID* を得る *(d* 方向*)*

- template<typename CUT_BID >
  void cutlib::GetCutBid (const CUT_BID *cb, size_t ijk, BidType bid[])
  一次元配列データから境界 *ID* を得る *(6* 方向まとめて*)*

変数

- const int32_t cutlib::CutPos8Clear0 [3]
  クリアマスク *(*方向別*)*

- const int32_t cutlib::CutPos8Clear = (int32_t)255 | ((int32_t)255<<8) | ((int32_t)255<<16)
  クリアマスク *(6* 方向まとめて*)*

- const int32_t cutlib::CutBid8Clear0 [3]
  クリアマスク *(*方向別*)*

- const int32_t cutlib::CutBid5Clear0 [6]
  クリアマスク *(*方向別*)*

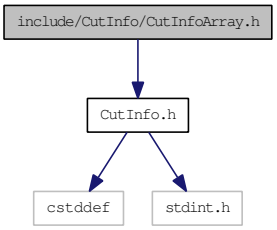
7.1.1 説明

交点情報基本データ型

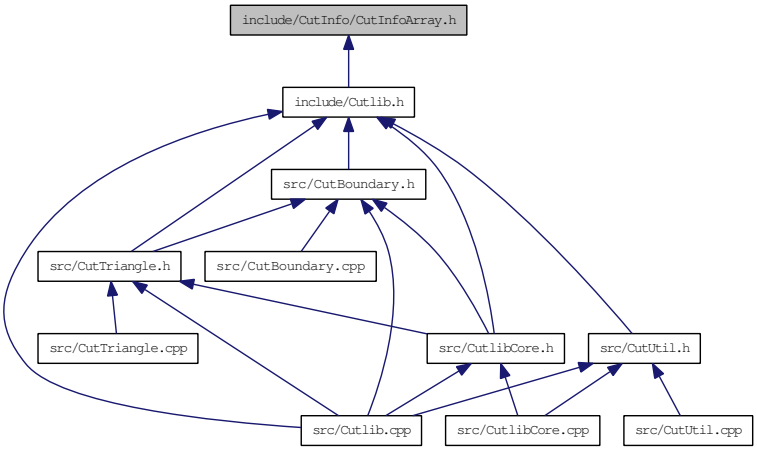
7.2 include/CutInfo/CutInfoArray.h

交点情報配列ラッパクラス#include "CutInfo.h"

CutInfoArray.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



構成

- class cutlib::CutInfoArray

  一次元配列ラッパクラスの基底クラス

- class cutlib::CutPosArray

  交点座標一次元配列ラッパ仮想クラス

- class cutlib::CutBidArray

  境界 *ID* 一次元配列ラッパ仮想クラス

- class cutlib::CutPosArrayTemplate< CUT_POS >

  交点座標一次元配列ラッパクラステンプレート

- class cutlib::CutBidArrayTemplate< CUT_BID >
  境界 *ID* **一次元配列ラッパクラステンプレート**

型定義

- typedef CutPosArrayTemplate< CutPos32 > cutlib::CutPos32Array
  CutPos32 **型交点座標配列ラッパクラス**.
- typedef CutPosArrayTemplate< CutPos8 > cutlib::CutPos8Array
  CutPos8 **型交点座標配列ラッパクラス**.
- typedef CutBidArrayTemplate< CutBid8 > cutlib::CutBid8Array
  CutBid8 **型境界** *ID* **配列ラッパクラス**.
- typedef CutBidArrayTemplate< CutBid5 > cutlib::CutBid5Array
  CutBid5 **型境界** *ID* **配列ラッパクラス**.

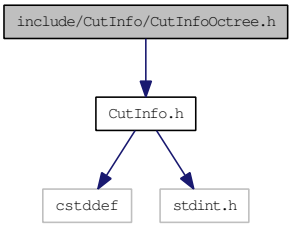
7.2.1 説明

交点情報配列ラッパクラス

7.3 include/CutInfo/CutInfoOctree.h

Octree セルデータへのアクセッサクラス. `#include "CutInfo.h"`

CutInfoOctree.h のインクルード依存関係図



構成

- class cutlib::CutPosOctree
  Octree 交点座標データアクセッサ仮想クラス.
- class cutlib::CutBidOctree
  Octree 境界 *ID* データアクセッサ仮想クラス.
- class cutlib::CutPosOctreeTemplate< CUT_POS, SIZE_IN_FLOAT >
  Octree 交点座標データアクセサクラステンプレート.
- class cutlib::CutBidOctreeTemplate< CUT_BID, SIZE_IN_FLOAT >
  Octree 境界 *ID* データアクセサクラステンプレート.

型定義

- typedef CutPosOctreeTemplate< CutPos32, 6 > cutlib::CutPos32Octree
  CutPos32 型交点座標データアクセッサクラス.
- typedef CutPosOctreeTemplate< CutPos8, 2 > cutlib::CutPos8Octree
  CutPos8 型交点座標データアクセッサクラス.
- typedef CutBidOctreeTemplate< CutBid8, 2 > cutlib::CutBid8Octree
  CutBid8 型境界 *ID* データアクセッサクラス.
- typedef CutBidOctreeTemplate< CutBid5, 1 > cutlib::CutBid5Octree
  CutBid5 型境界 *ID* データアクセッサクラス.

7.3.1 説明

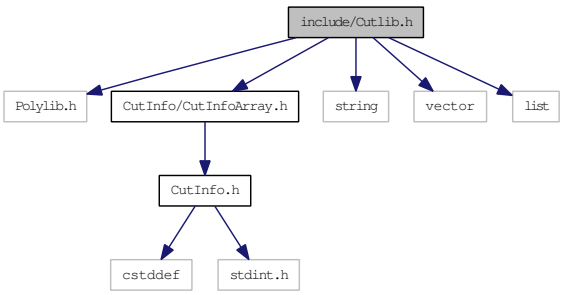
Octree セルデータへのアクセッサクラス.

7.4 include/Cutlib.h

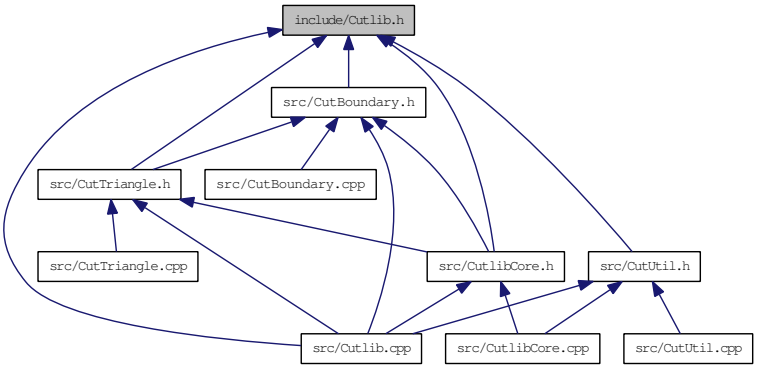
境界情報計算関数 宣言#include "Polylib.h"

```
#include "CutInfo/CutInfoArray.h"
#include <string>
#include <vector>
#include <list>
```

Cutlib.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



型定義

- typedef std::vector< Triangle * > cutlib::Triangles
  Polylib **検索結果格納リスト**.

列挙型

- enum cutlib::CutlibReturn {

cutlib::CL_SUCCESS = 0, cutlib::CL_BAD_GROUP_LIST = 1, cutlib::CL_BAD_POLYLIB = 2, cutlib::CL_BAD_SKLTREE = 3, cutlib::CL_SIZE_EXCEED = 4, cutlib::CL_OTHER_ERROR = 10 }

交点情報計算関数リターンコード

関数

- CutlibReturn cutlib::CutInfoCell (const size_t ista[], const size_t nlen[], const float org[], const float d[], const Polylib *pl, CutPosArray *cutPos, CutBidArray *cutBid)

  交点情報計算: **セル中心間**, **配列ラッパクラス**, **計算領域指定**

- CutlibReturn cutlib::CutInfoCell (const float org[], const float d[], const Polylib *pl, CutPosArray *cutPos, CutBidArray *cutBid)

  交点情報計算: **セル中心間**, **配列ラッパクラス**, **全領域**

- CutlibReturn cutlib::CutInfoNode (const size_t ista[], const size_t nlen[], const float org[], const float d[], const Polylib *pl, CutPosArray *cutPos, CutBidArray *cutBid)

  交点情報計算: **ノード間**, **配列ラッパクラス**, **計算領域指定**

- CutlibReturn cutlib::CutInfoNode (const float org[], const float d[], const Polylib *pl, CutPosArray *cutPos, CutBidArray *cutBid)

  交点情報計算: **ノード間**, **配列ラッパクラス**, **全領域**

- template<typename CUT_POS , typename CUT_BID >
  CutlibReturn cutlib::CutInfoCell (const size_t ndim[], const size_t ista[], const size_t nlen[], const float org[], const float d[], const Polylib *pl, CUT_POS cutPos[], CUT_BID cutBid[])

  交点情報計算: **セル中心間**, **一次元配列**, **計算領域指定**

- template<typename CUT_POS , typename CUT_BID >
  CutlibReturn cutlib::CutInfoCell (const size_t ndim[], const float org[], const float d[], const Polylib *pl, CUT_POS cutPos[], CUT_BID cutBid[])

  交点情報計算: **セル中心間**, **一次元配列**, **全領域**

- template<typename CUT_POS , typename CUT_BID >
  CutlibReturn cutlib::CutInfoNode (const size_t ndim[], const size_t ista[], const size_t nlen[], const float org[], const float d[], const Polylib *pl, CUT_POS cutPos[], CUT_BID cutBid[])

  交点情報計算: **ノード間**, **一次元配列**, **計算領域指定**

- template<typename CUT_POS , typename CUT_BID >
  CutlibReturn cutlib::CutInfoNode (const size_t ndim[], const float org[], const float d[], const Polylib *pl, CUT_POS cutPos[], CUT_BID cutBid[])

交点情報計算: ノード間, 一次元配列, 全領域

7.4.1 説明

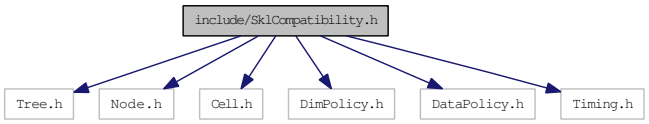
境界情報計算関数 宣言

7.5 include/SklCompatibility.h

pFTT クラスライブラリとの互換性のためのヘッダ`#include "Tree.h"`

```
#include "Node.h"
#include "Cell.h"
#include "DimPolicy.h"
#include "DataPolicy.h"
#include "Timing.h"
```

SklCompatibility.h のインクルード依存関係図



型定義

- typedef TD_NAMESPACE::Tree< TD_NAMESPACE::ThreeDimPolicy, TD_NAMESPACE::DefaultDataPolicy< float > > **SklTree**
- typedef SklTree::cell_type **SklCell**
- typedef SklCell::NeighborSet **SklNghbrSet**
- typedef SklTree::SplitList **SklSplitList**

関数

- double **SklGetTime** ()

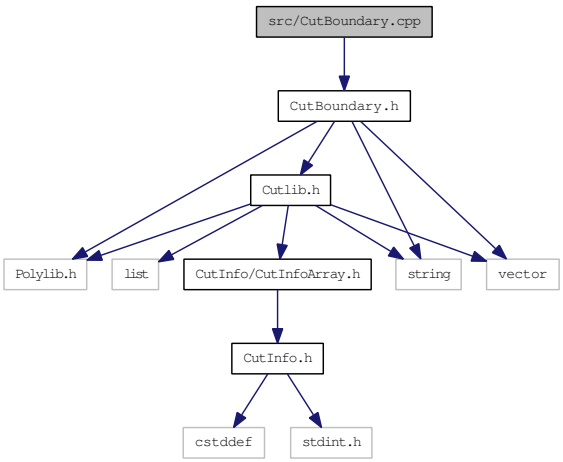
7.5.1 説明

pFTT クラスライブラリとの互換性のためのヘッダ Compatibility.h の内容を「Sph〜」を「Skl〜」に修正して作成

7.6 src/CutBoundary.cpp

Polylib ポリゴングループ--境界 ID 対応付けクラス 実装. `#include "CutBoundary.h"`

CutBoundary.cpp のインクルード依存関係図



7.6.1 説明

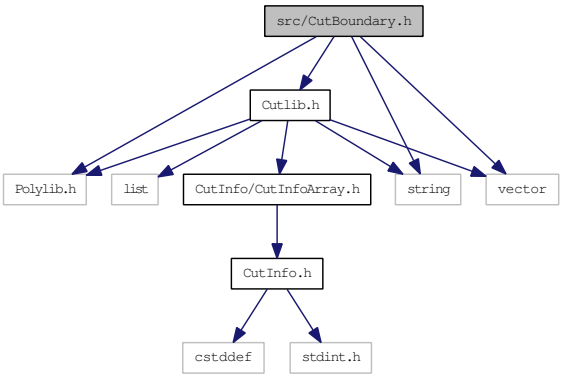
Polylib ポリゴングループ--境界 ID 対応付けクラス 実装.

7.7 src/CutBoundary.h

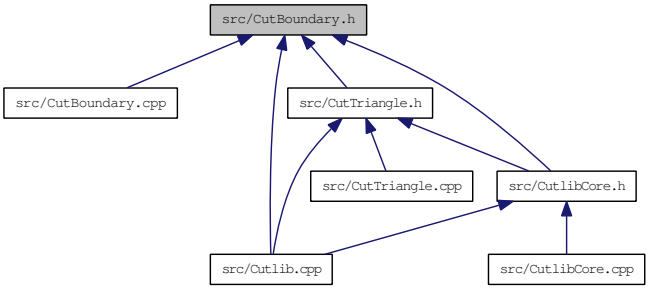
Polylib ポリゴングループ--境界 ID 対応付けクラス 宣言. `#include "Cutlib.h"`

```
#include "Polylib.h"
#include <string>
#include <vector>
```

CutBoundary.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



構成

- class cutlib::CutBoundary

  Polylib **ポリゴングループ**--**境界** *ID* **対応付けクラス**.

型定義

- typedef std::vector< CutBoundary > cutlib::CutBoundaries

Polylib **ポリゴングループ--境界** *ID* **対応リスト**.

7.7.1 説明

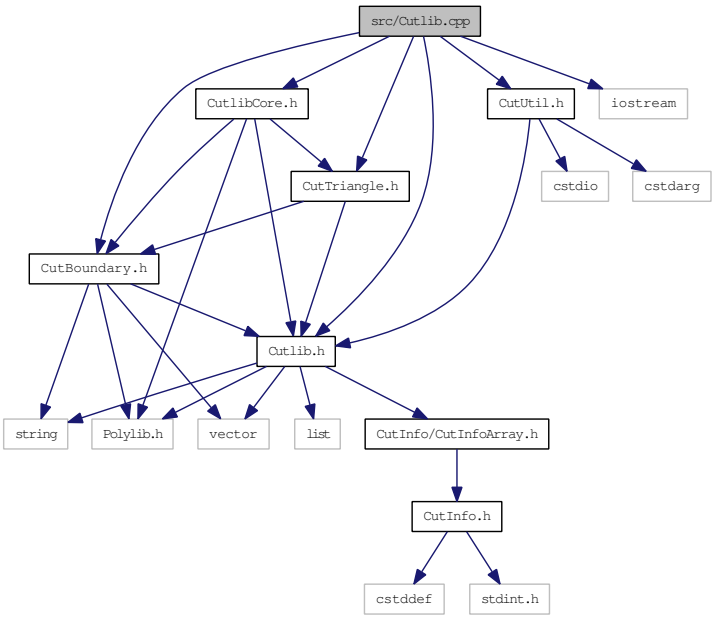
Polylib ポリゴングループ--境界 ID 対応付けクラス 宣言.

7.8 src/Cutlib.cpp

境界情報計算関数 実装#include "Cutlib.h"

```
#include "CutlibCore.h"
#include "CutUtil.h"
#include "CutTriangle.h"
#include "CutBoundary.h"
#include <iostream>
```

Cutlib.cpp のインクルード依存関係図



関数

- CutlibReturn cutlib::CutInfoCell (const size_t ista[], const size_t nlen[], const float org[], const float d[], const Polylib *pl, CutPosArray *cutPos, CutBidArray *cutBid)

  交点情報計算: セル中心間, 配列ラッパクラス, 計算領域指定

- CutlibReturn cutlib::CutInfoNode (const size_t ista[], const size_t nlen[], const float org[], const float d[], const Polylib *pl, CutPosArray *cutPos, CutBidArray *cutBid)

  交点情報計算: ノード間, 配列ラッパクラス, 計算領域指定

- CutlibReturn cutlib::CutInfoCell (const float org[], const float d[], const Polylib *pl, CutPosArray *cutPos, CutBidArray *cutBid)

交点情報計算: **セル中心間**, **配列ラッパクラス**, **全領域**

- CutlibReturn cutlib::CutInfoNode (const float org[], const float d[], const Polylib *pl, CutPosArray *cutPos, CutBidArray *cutBid)

  交点情報計算: **ノード間**, **配列ラッパクラス**, **全領域**

7.8.1 説明

境界情報計算関数 実装

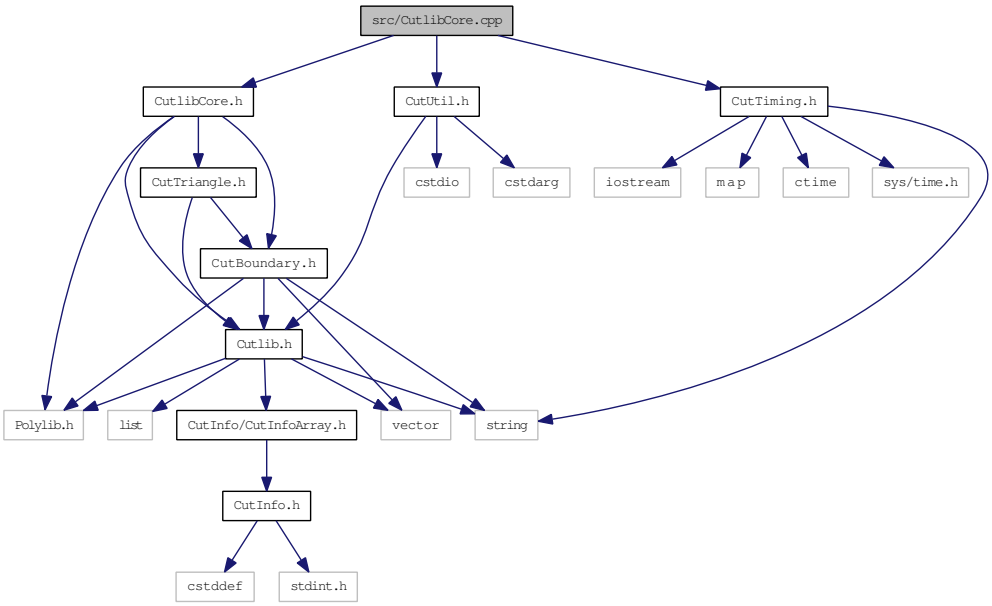
7.9 src/CutlibCore.cpp

境界情報計算関数 (コア部分) 実装#include "CutlibCore.h"

#include "CutUtil.h"

#include "CutTiming.h"

CutlibCore.cpp のインクルード依存関係図



列挙型

- enum

関数

- void cutlib::core::calcCutInfo (const Vec3f &c, const Vec3f &d, const Polylib *pl, const CutBoundaries *bList, float pos6[], BidType bid6[])

  指定された計算点 (セル中心 *or* ノード) で交点情報を計算

7.9.1 説明

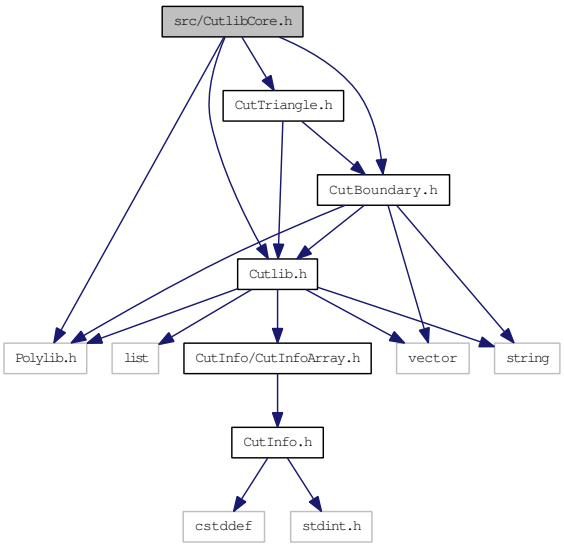
境界情報計算関数 (コア部分) 実装

7.10 src/CutlibCore.h

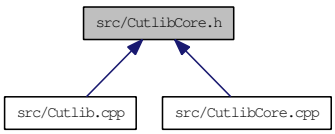
境界情報計算関数 (コア部分) 宣言`#include "Cutlib.h"`

```
#include "CutBoundary.h"
#include "CutTriangle.h"
#include "Polylib.h"
```

CutlibCore.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



関数

- void cutlib::core::calcCutInfo (const Vec3f &c, const Vec3f &d, const Polylib *pl, const CutBoundaries *bList, float pos6[], BidType bid6[])

  指定された計算点 (セル中心 *or* ノード) で交点情報を計算

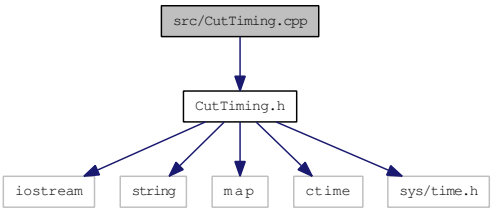
7.10.1 説明

境界情報計算関数 (コア部分) 宣言

7.11 src/CutTiming.cpp

時間測定用クラス 実装#include "CutTiming.h"

CutTiming.cpp のインクルード依存関係図



関数

- ostream & **cutlib::operator**<< (ostream &os, const Timing &t)

7.11.1 説明

時間測定用クラス 実装

7.12 src/CutTiming.h

時間測定用クラス 宣言#include <iostream>

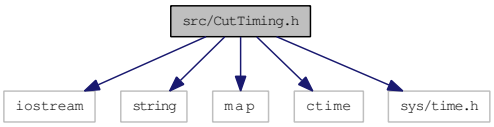
#include <string>

#include <map>

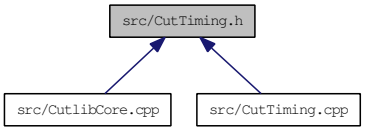
#include <ctime>

#include <sys/time.h>

CutTiming.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



構成

- class cutlib::Timing
  時間測定ストップウオッチクラス

7.12.1 説明

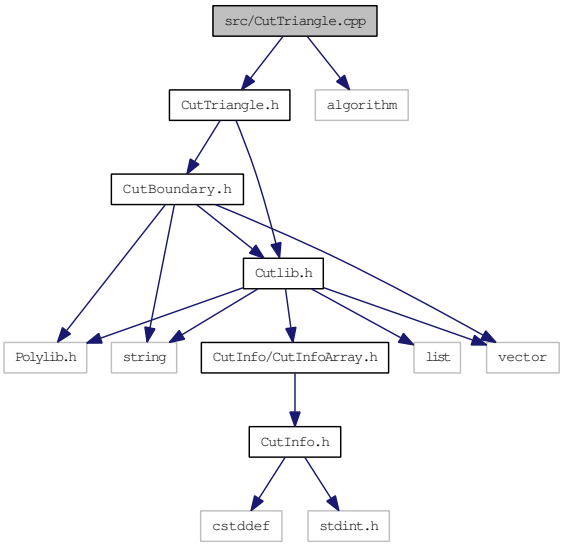
時間測定用クラス 宣言

7.13 src/CutTriangle.cpp

カスタムポリゴンリスト用クラス 実装#include "CutTriangle.h"

#include <algorithm>

CutTriangle.cpp のインクルード依存関係図



列挙型

- enum

7.13.1 説明

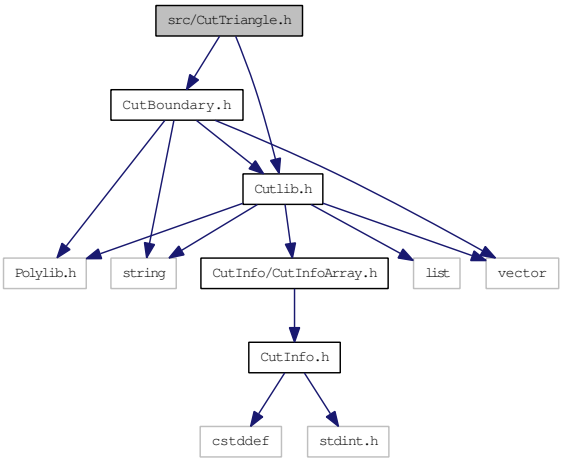
カスタムポリゴンリスト用クラス 実装

7.14 src/CutTriangle.h

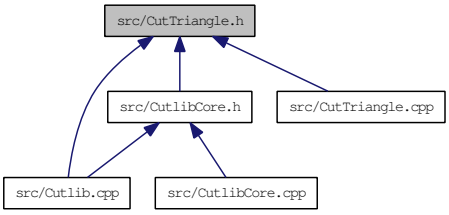
カスタムポリゴンリスト用クラス 宣言`#include "Cutlib.h"`

`#include "CutBoundary.h"`

CutTriangle.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



構成

- class cutlib::CutTriangle

  BBox(binding box) 情報と境界 *ID* を持つカスタムポリゴンクラス.

型定義

- typedef std::vector< CutTriangle * > cutlib::CutTriangles

  三角形リスト

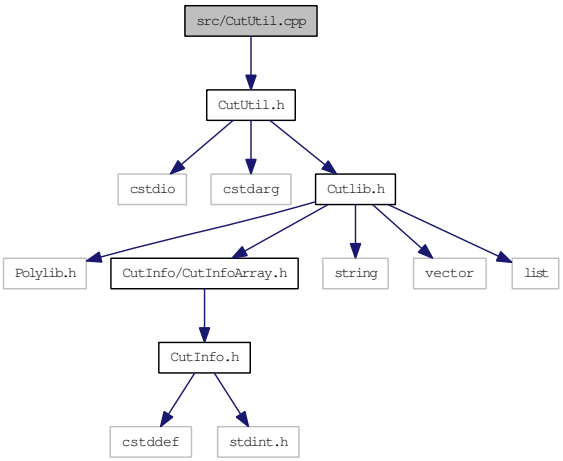
7.14.1 説明

カスタムポリゴンリスト用クラス 宣言

7.15 src/CutUtil.cpp

補助関数群 実装#include "CutUtil.h"

CutUtil.cpp のインクルード依存関係図



列挙型

- enum

関数

- bool cutlib::util::intersectXY (const Triangle *t, float x, float y, float &z)
  三角形と z 軸に平行な直線との交点を計算
- bool cutlib::util::intersectYZ (const Triangle *t, float y, float z, float &x)
  三角形と x 軸に平行な直線との交点を計算
- bool cutlib::util::intersectZX (const Triangle *t, float z, float x, float &y)
  三角形と y 軸に平行な直線との交点を計算
- void cutlib::util::errMsg (const char *fmt,...)
  エラーメッセージ出力

7.15.1 説明

補助関数群 実装

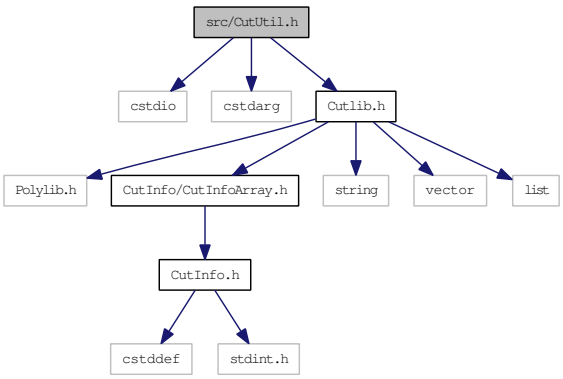
7.16 src/CutUtil.h

補助関数群 宣言#include <cstdio>

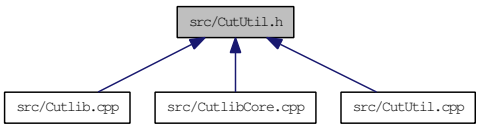
#include <cstdarg>

#include "Cutlib.h"

CutUtil.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



関数

- void cutlib::util::errMsg (const char *fmt,...)

  エラーメッセージ出力

- bool cutlib::util::intersectXY (const Triangle *t, float x, float y, float &z)

  三角形と z 軸に平行な直線との交点を計算

- bool cutlib::util::intersectYZ (const Triangle *t, float y, float z, float &x)

  三角形と x 軸に平行な直線との交点を計算

- bool cutlib::util::intersectZX (const Triangle *t, float z, float x, float &y)

  三角形と y 軸に平行な直線との交点を計算

7.16.1 説明

補助関数群 宣言

Index