SoftimageSDK入門

COLLADAエクスポーターを作ってみよう

株式会社　セガ
AM開発技術部
立福　寛
Tatefuku_Hiroshi@sega.co.jp

このセッションの目的

- Softimageを使っているけどプラグインの作り方がよくわからない・・・
- **SDKの情報が少ない**
  - ドキュメントが足りない
  - サンプルが少ない
- **エクスポーターの作り方の情報も少ない**
  - 開発者が少ない
  - 講演・セミナーが少ない

このセッションの目的

- 要するに・・・**情報が全然足りない**

  → **ここをカバーするのが今回の目的**

このセッションで得られること

- SoftimageSDKの基本的な使い方
- エクスポーターの作り方に関するノウハウ

セッションの資料について

- プレゼン資料（１４５ページぐらい）
- 更に詳細なドキュメント（pdfで８５ページ）
- エクスポーター本体のソースコード
  - フルスクラッチ（依存モジュールなし）
  - 説明した機能はフル実装
  - COLLADA準拠
  - VS2008, Softimage2010向け

このセッションについて

- 資料に書いてある内容の**メモは不要**
- このセッションは結構**早い**です
- 話を聞くことに**集中してください**

- デモはありません
- 質疑応答の時間もありません

ドキュメントについて

- 今回の発表内容は**配布用のドキュメント**からの抜粋です
- わからなかったところは後で**ドキュメントを参照してください**（詳しく説明しています）
- ソースも全部付いているので安心

注意点

- 今回はSoftimageSDKとエクスポーターの話をします
- **COLLADAの話はほとんどありません**

  → 興味のある方はソースを参照

本日の内容一覧

- SoftimageSDKの基本
- エクスポーターの概要
- ノード
- ジオメトリ
- マテリアル
- アニメーション
- スキニング
- シェイプ
- カスタムパラメータ
- エクスポーター作成のコツ

SoftimageSDKの基本

- SDKの概要
- プラグインのデバッグ（環境設定）
- 最初のSoftimageプラグイン
  - HelloWorldプラグイン

～ SoftimageSDKの基本 ～

SDKの特徴

- COM/C++API、各種スクリプト言語によるプラグイン開発が可能
  - COM/C++API　→　dll
  - スクリプト言語　→　*.vbs, *.py, …
- 今回は**C++API**と**Pythonスクリプト**を使用

～ SoftimageSDKの基本 ～

プラグインのデバッグ

- プラグインはデバッグできるの？
  →**VisualStudioからデバッグ可能**

- 適切な**環境設定**が必要
  →　詳細はドキュメントを参照

～ SoftimageSDKの基本 ～

最初のSoftimageプラグイン

- HelloWorldプラグイン
- スクリプトエディタに”Hello World!”と表示
- C++APIで作成(dll)

～ SoftimageSDKの基本 ～

プラグインの基本関数

- 特定の名前の関数を定義
- コマンド名がHelloWorldの場合

ロード：XSILoadPlugin()
アンロード：XSIUnloadPlugin()
初期化：HelloWorld_Init( )
実行：HelloWorld_Execute()

- 詳細はサンプルを参照

～ SoftimageSDKの基本 ～

プラグイン（dll）のビルド

1. 環境設定
2. プロジェクトファイルの設定
3. 関数を作成
4. ビルド
5. プラグイン（**dll**）が完成

～ SoftimageSDKの基本 ～

プラグインの実行

1. dllをユーザールートにコピー
2. Softimageを起動　→　自動ロード
3. **プラグインマネージャーでロードを確認**
4. **スクリプトエディタで実行**

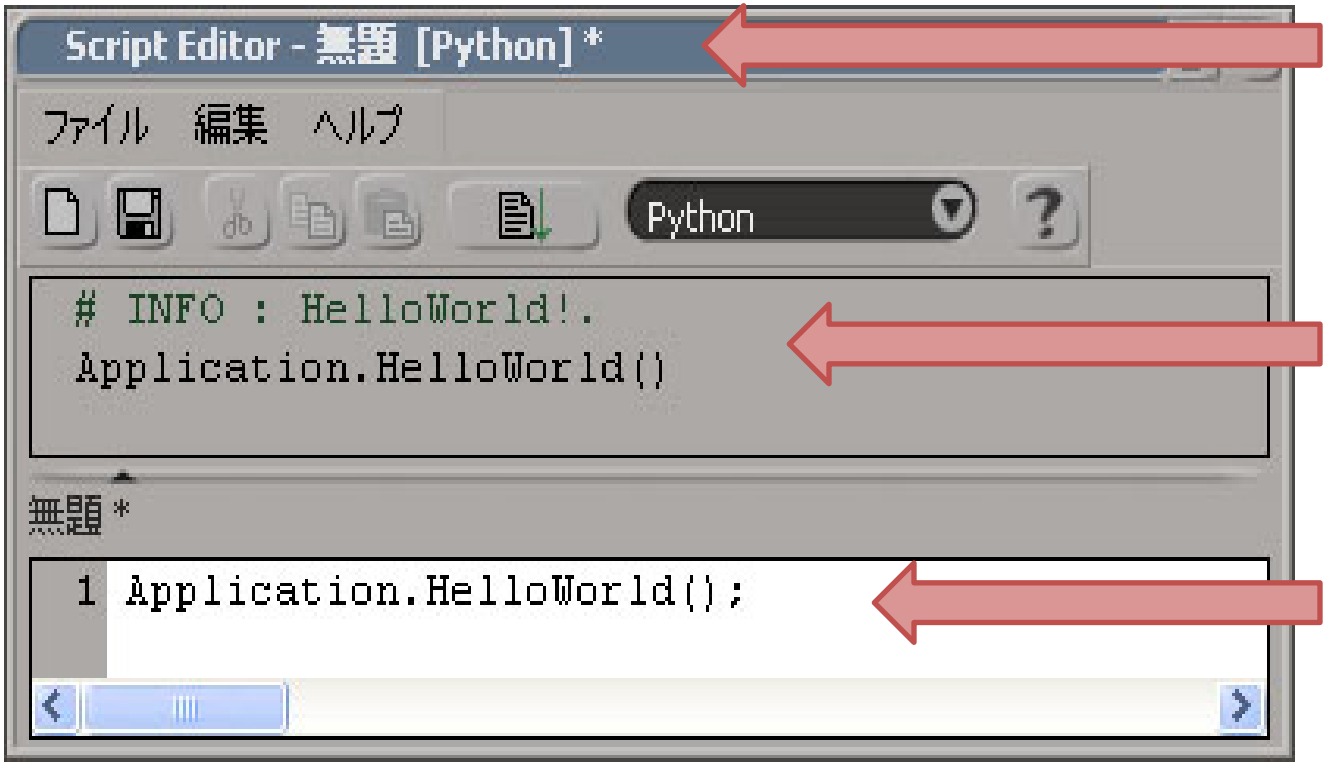
～ SoftimageSDKの基本 ～

プラグインマネージャー

- プラグインの状態を確認するツール
- dllが正常にロードできたかチェック
- 非常に便利

～ SoftimageSDKの基本 ～
スクリプトエディタ

- スクリプトを実行するウィンドウ



スクリプト言語
Python

実行結果

コマンド
（F5で実行）

～ SoftimageSDKの基本 ～

まとめ

- SDKの概要
- プラグインのデバッグ（環境設定）
- 最初のSoftimageプラグイン
  - HelloWorldプラグイン

エクスポーターの概要

- サポートする要素
- エクスポーターの全体構成
- C++API dll
- COLLADAライブラリ

～エクスポーターの概要～

サポートする要素

- ノード
- ジオメトリ
- マテリアル
- アニメーション
- スキニング
- シェイプ
- カスタムパラメータ

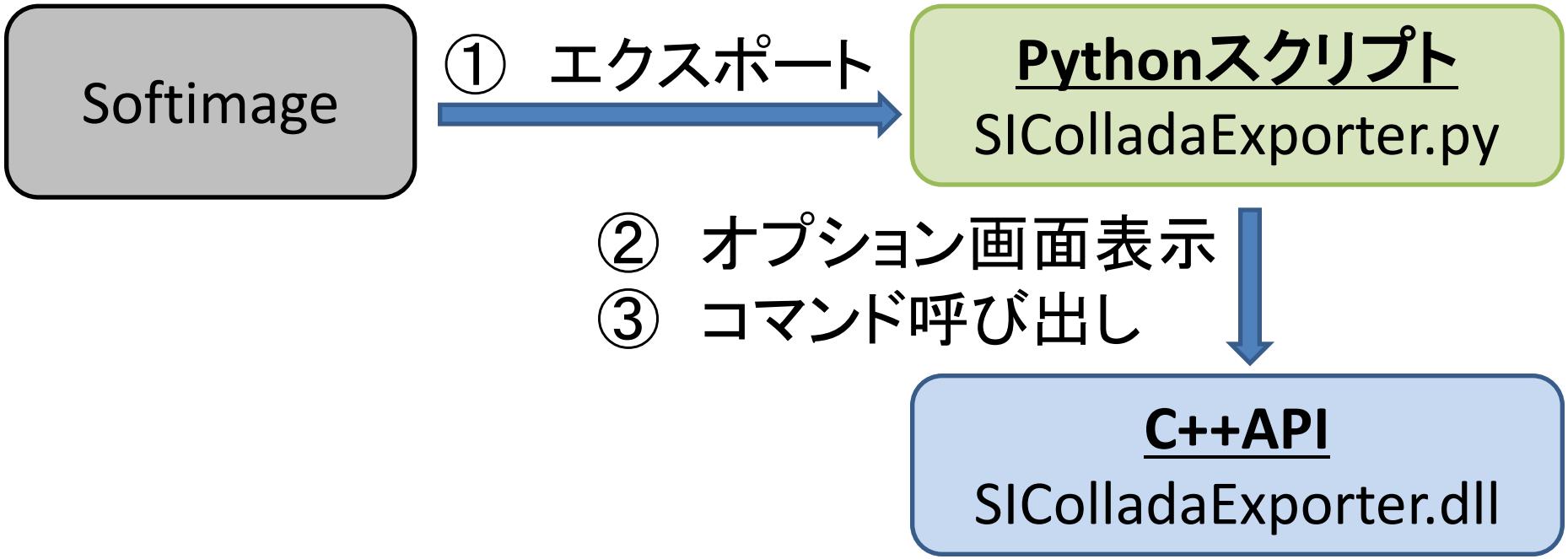
→ **ゲーム開発に必要なものを一通り出力**

～エクスポーターの概要～

エクスポーターの全体構成

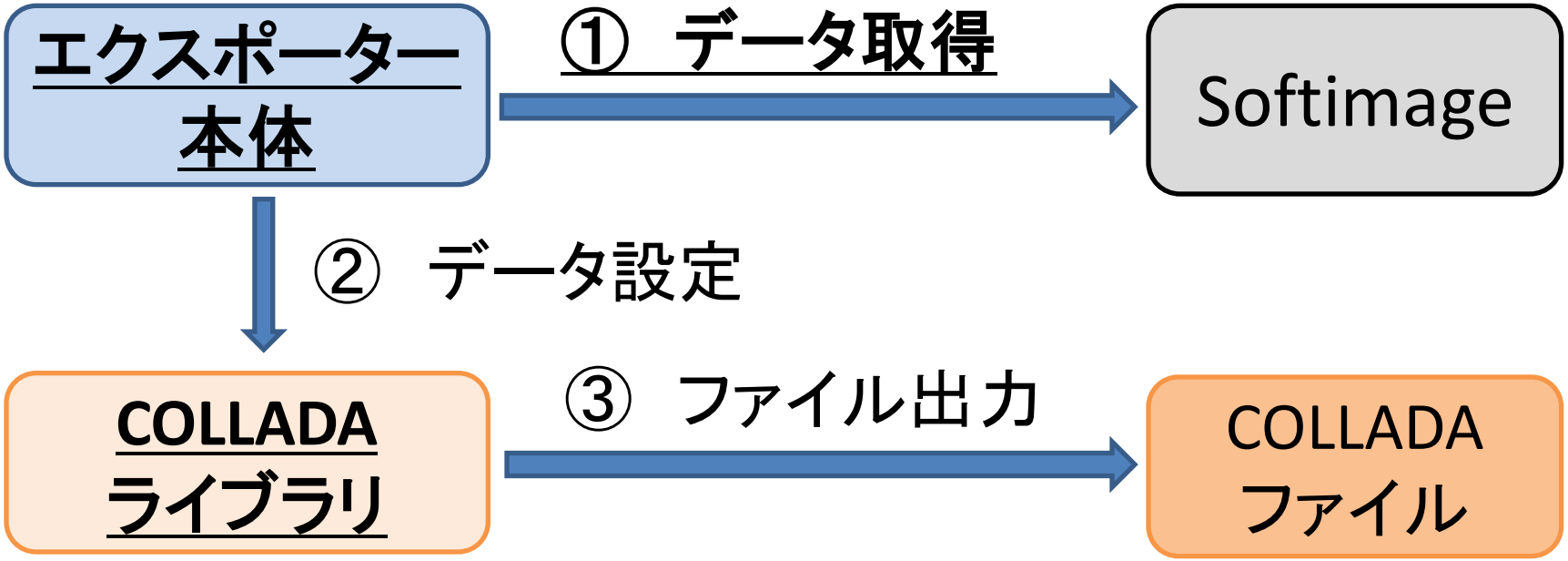
- エクスポーター本体：**C++API dll**
- メニュー/GUI：**Pythonスクリプト**

（社内ツールと同じ構成）



～エクスポーターの概要～
C++API dll

- C++API dllの内部構成
  - エクスポーター本体
  - COLLADAライブラリ



～エクスポーターの概要～
COLLADAライブラリ

- 公式COLLADA DOM　→　大きい、使いにくい
- **独自作成のCOLLADAライブラリ**（ソース付き）

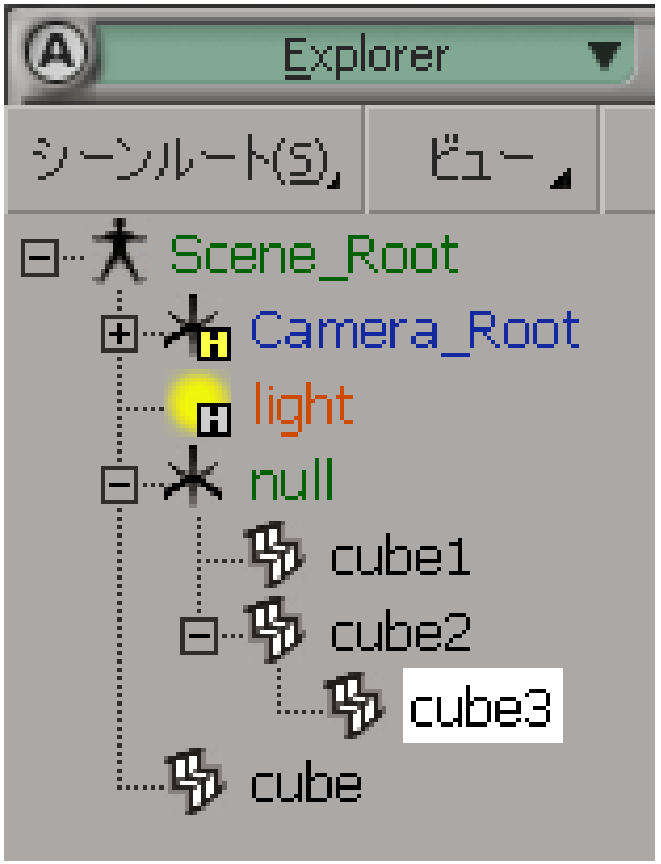
～エクスポーターの概要～

まとめ

- サポートする要素
- エクスポーターの全体構成
- C++API dll
- COLLADAライブラリ

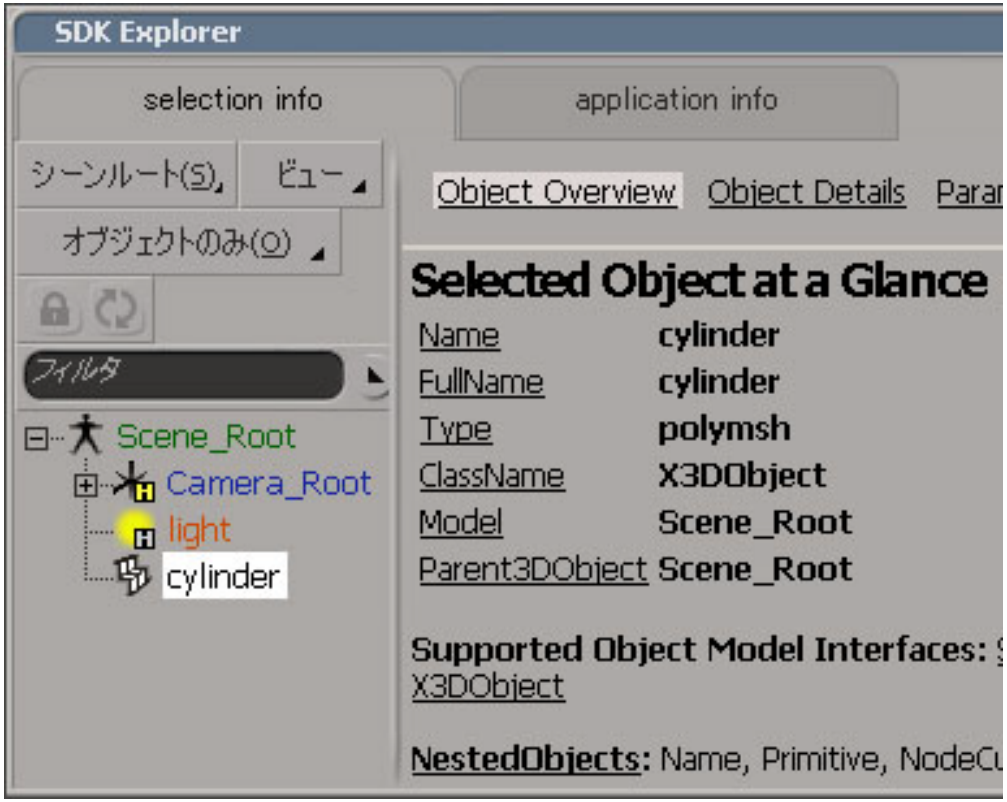
ノード情報の取得

- SDKエクスプローラー
- 選択/非選択
- 名前
- ノードの種類
- 親子関係
- 座標



～ノード情報の取得～
SDKエクスプローラー

- オブジェクトの中身をのぞくツール
- **デバッグ用**（プログラムを書くより早い）



～ノード情報の取得～
選択/非選択

- エクスポーターの出力モード
  - シーン全体
  - **選択したもの**

→ **選択状態の取得**

～ノード情報の取得～
選択/非選択

- **Application::GetSelection**()
- **Applicationクラス　＝　アプリケーション(Softimage)**

```
Application app;
CRefArray selection = app.GetSelection();
```

- 結果は**CRefArrayクラス**

→ **CRefクラス**の配列

～ノード情報の取得～
CRefクラス

- CRefクラスは**各種オブジェクトへの参照型**
- 適切なクラスへ**キャスト**して使う
  - CBase::IsValid()

～ノード情報の取得～
CBase::IsValid()

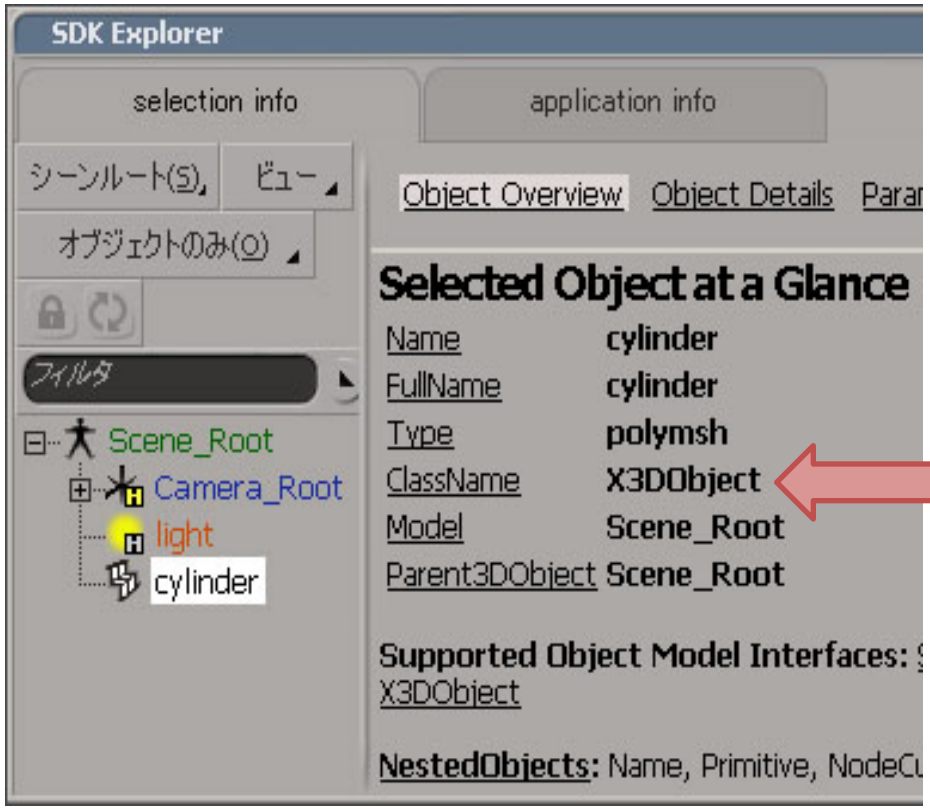
- あるクラスへの**変換が正しいか**チェック

```
CRef ref;
…
// ポリゴンメッシュ？
PolygonMesh polygonMesh( ref );
if( polygonMesh.IsValid() ) {
        …
}
```

～ノード情報の取得～

CBase::IsValid()

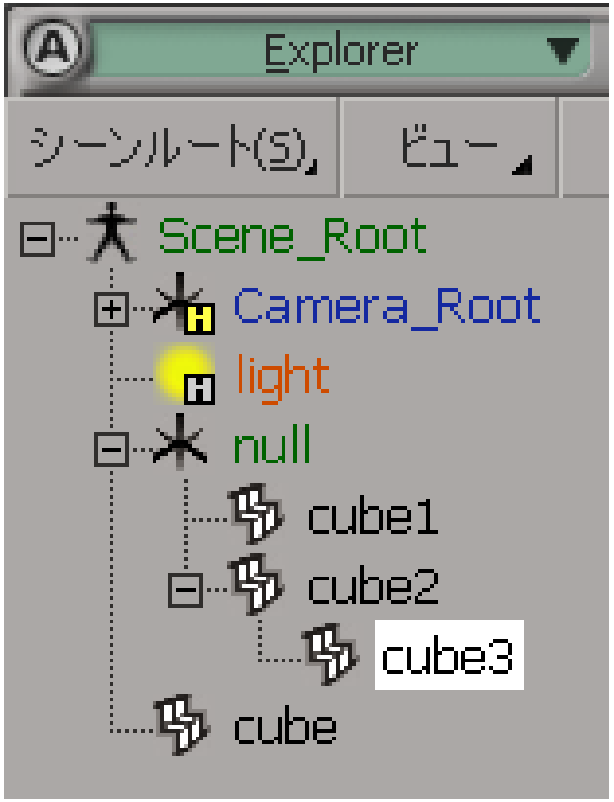
- このクラスにキャストできるか？
- **SDKエクスプローラー**でチェック



X3DObjectクラス

～ノード情報の取得～
名前

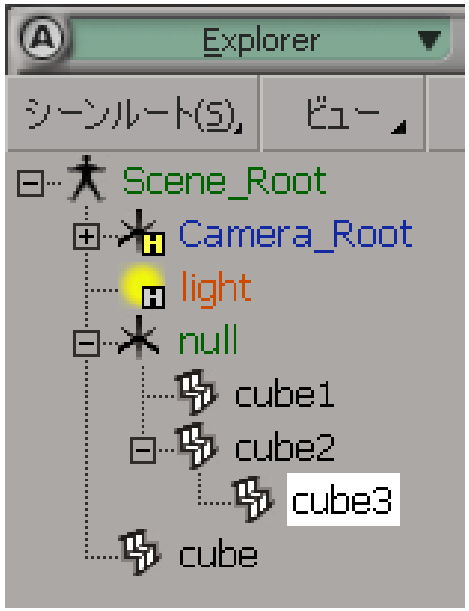
- SIObject::**GetName**()
- Softimage上の名前




```
SIObject siObject( ref );
CString name = siObject.GetName();
```

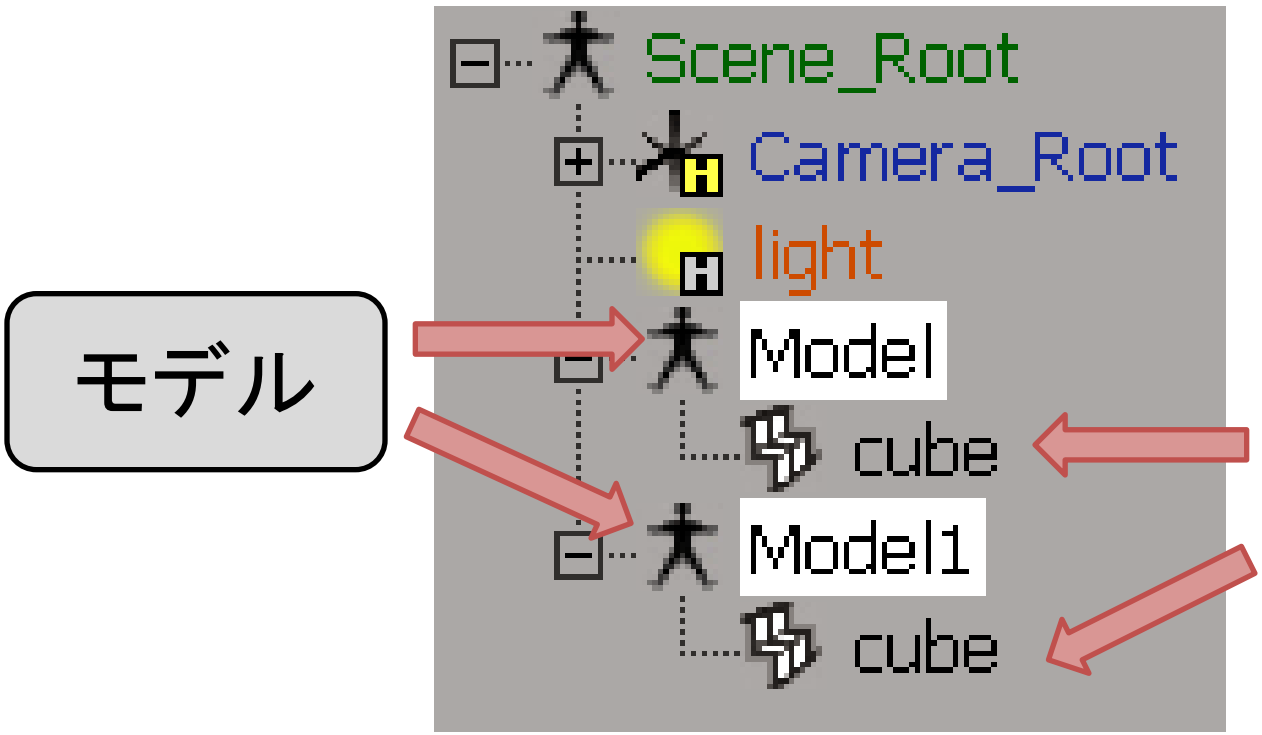
～ノード情報の取得～
ユニークな名前

- **ユニーク(唯一)な名前**がほしい
- 通常Softimageのノード名は**重複できない**
（別階層でも重複できない）



～ノード情報の取得～
ユニークな名前

- **モデル**：特殊なノード
- モデルの子供は**重複が許される**

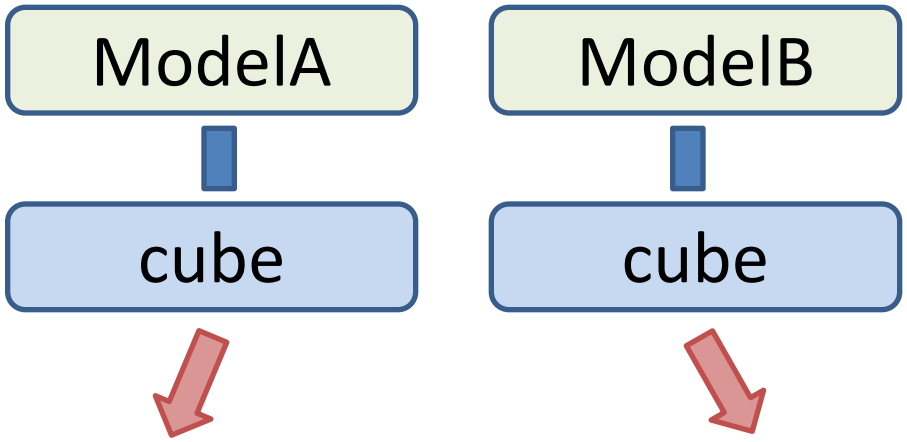


～ノード情報の取得～
ユニークな名前

- Softimageのノードの名前は**重複する**
- GetName()の名前は**そのまま使えない**

～ノード情報の取得～
ユニークな名前

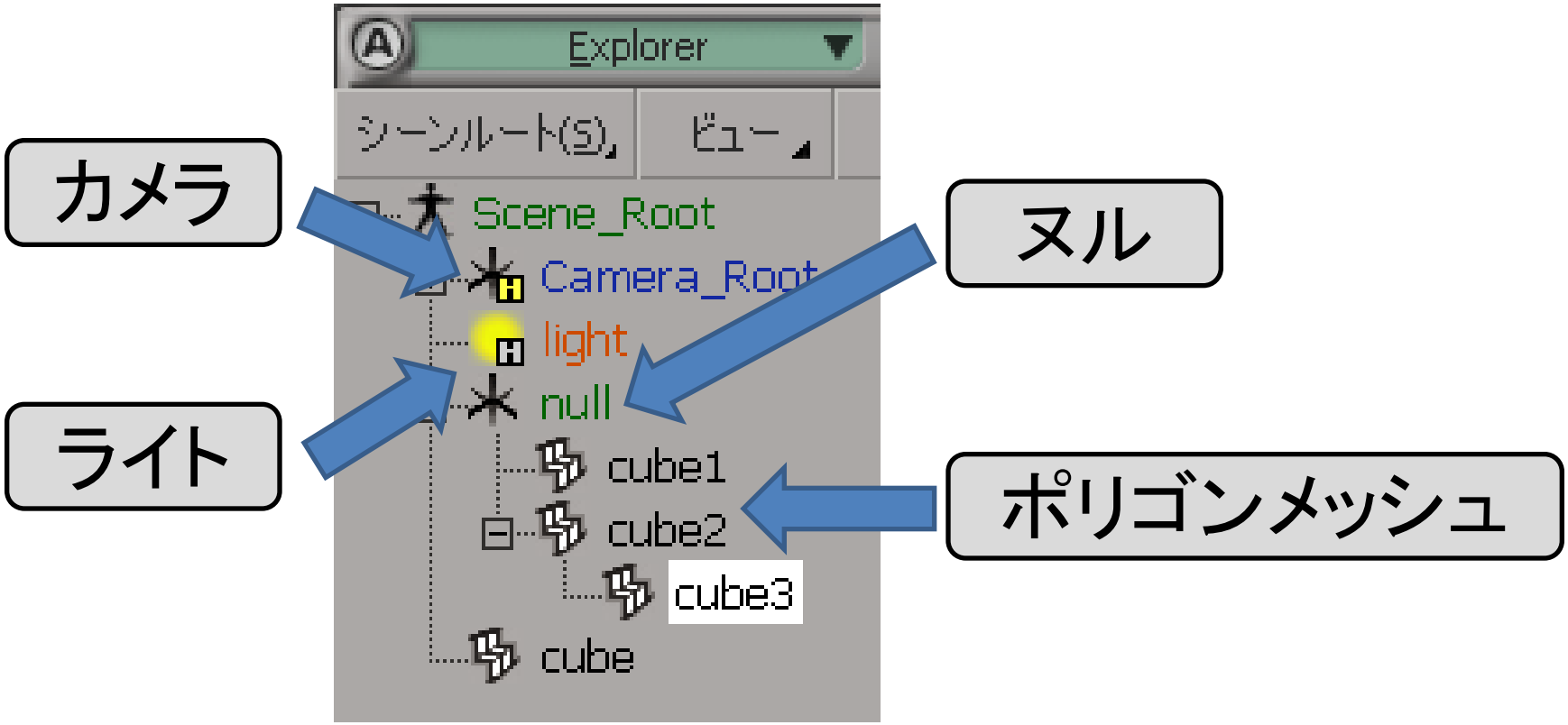
- **親子の名前を連結**



- ユニークな名前が手に入った

～ノード情報の取得～
ノードの種類

- **ノードの種類**を調べよう



～ノード情報の取得～

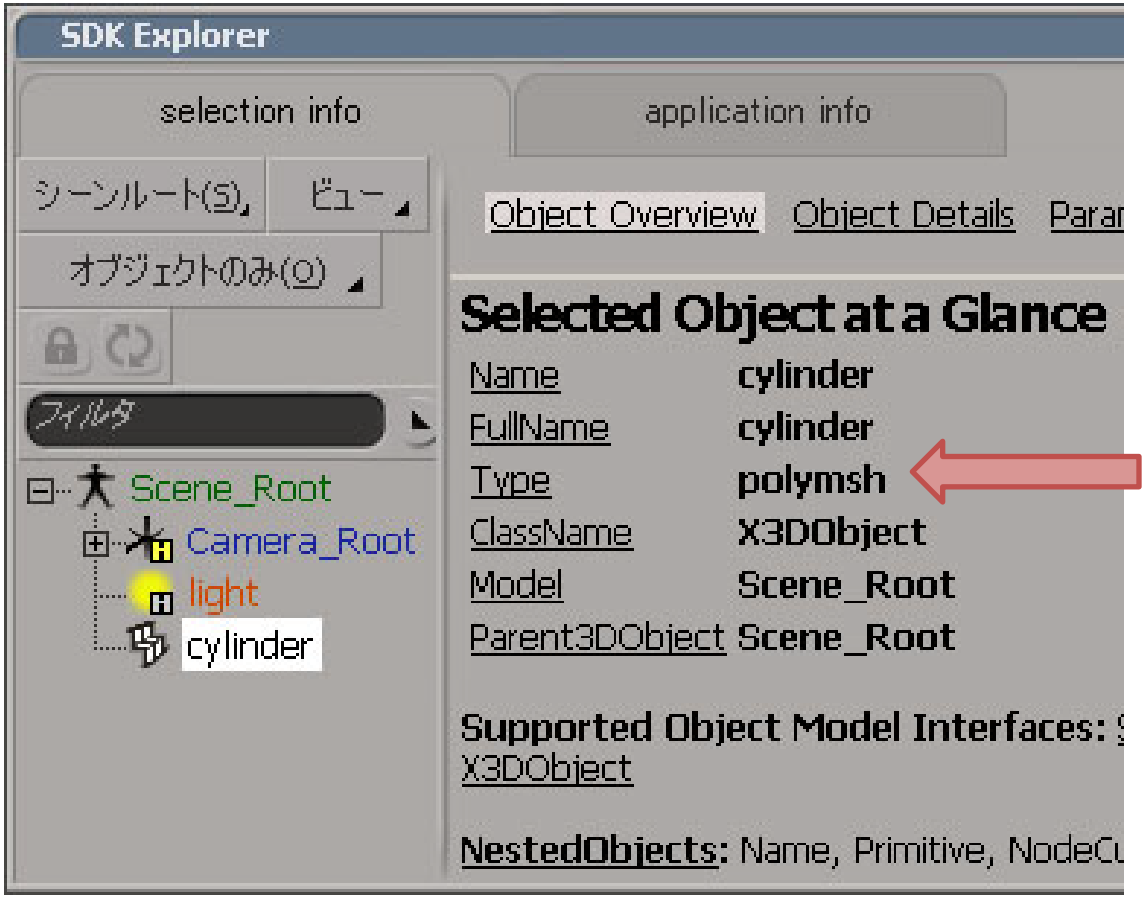
ノードの種類

- SceneItem::**GetType**()で取得
- 結果は文字列

```
SceneItem sceneItem( ref );
CString type = sceneItem.GetType();
if( type == L"camera" ) {
    /* カメラ情報を取得 */
}
else if( type == L"polymsh" ) {
    /* ポリゴンメッシュ情報を取得 */
}
```

～ノード情報の取得～
ノードの種類

- TypeはSDKエクスプローラーでチェック



“polymsh”

～ノード情報の取得～
親子関係

- ノードは**親子構造**になっている
- SIObject::**GetParent**で親ノードを取得
- 親がいない場合は自分自身を返す

```
SIObject parent = siObject.GetParent();
if( parent == siObject ) {
    /* 親がいない */
}
else {
    /* 親が存在 */
}
```

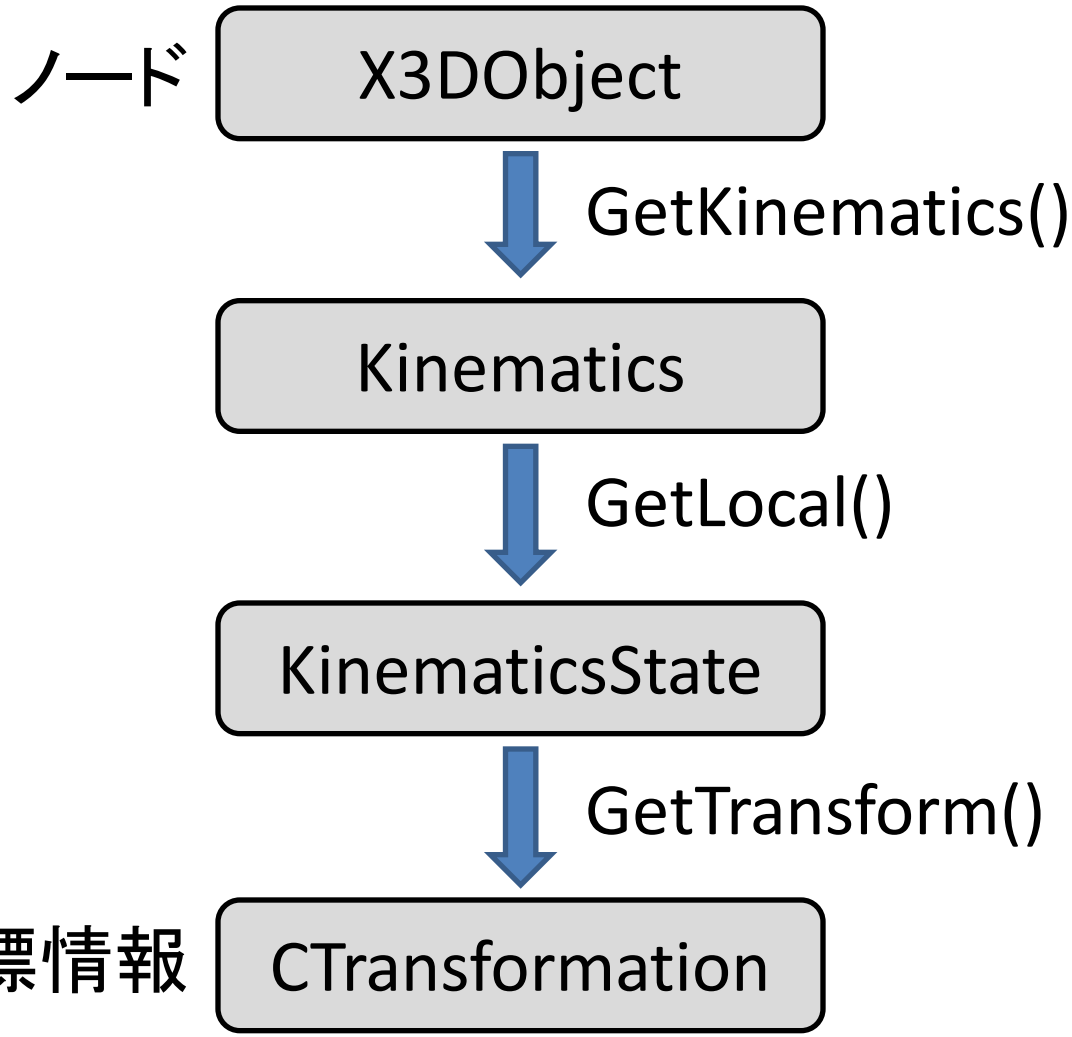
～ノード情報の取得～
座標

- 座標は**X3DObjectクラス**から取得

```
CRef ref;
…
X3DObject x3DObject( ref );
```

～ノード情報の取得～

X3DObject → CTransformation



～ノード情報の取得～
CTransformationクラス

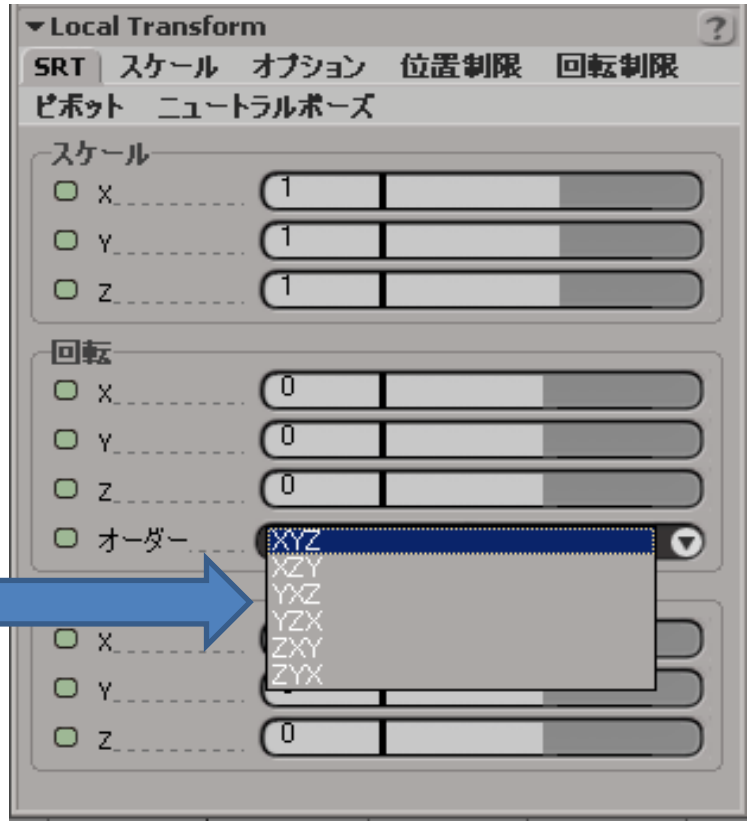
- **CTransformationクラス**から座標を取得

```
CVector3 s = trs.GetScaling();
CVector3 r = trs.GetRotationXYZAngles();
CVector3 t = trs.GetTranslation();
CQuaternion q = trs.GetRotationQuaternion();
```

～ノード情報の取得～
回転の順序

- 回転（XYZ）の順序を変更できる（6種類）
- 大半のDCCツールで可能

回転の
適用順序



～ノード情報の取得～

回転の順序

- 常にXYZとは限らない
- 順序に注意
- ランタイムで対応する？

→ デザイナーと相談

（XYZ固定でいいと思います・・・）

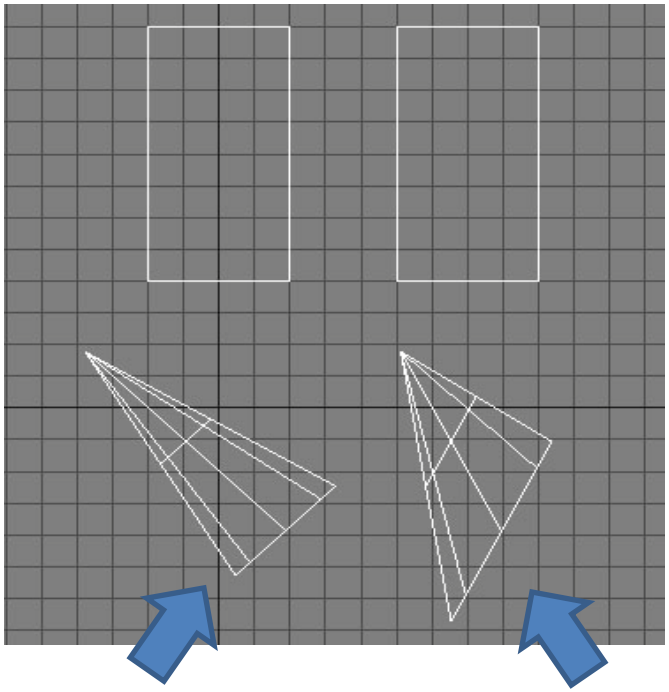
～ノード情報の取得～

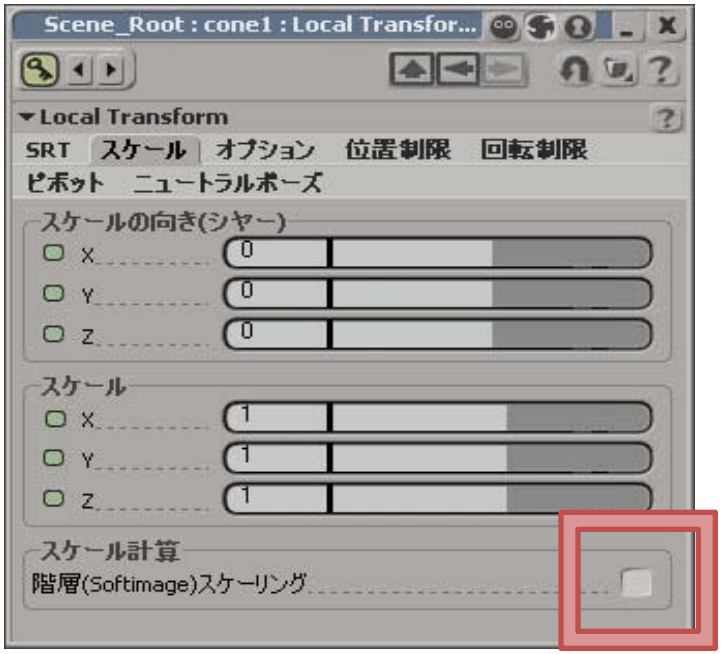
階層（Softimage）スケーリング

- Softimage特有のスケール計算に要注意

スケール
Y=2

回転
Z=45°





階層スケーリング**ON**　　階層スケーリング**OFF**

～ノード情報の取得～

階層（Softimage）スケーリング

- スケールだけ別計算
- ランタイムで対応する？

→　デザイナーと相談

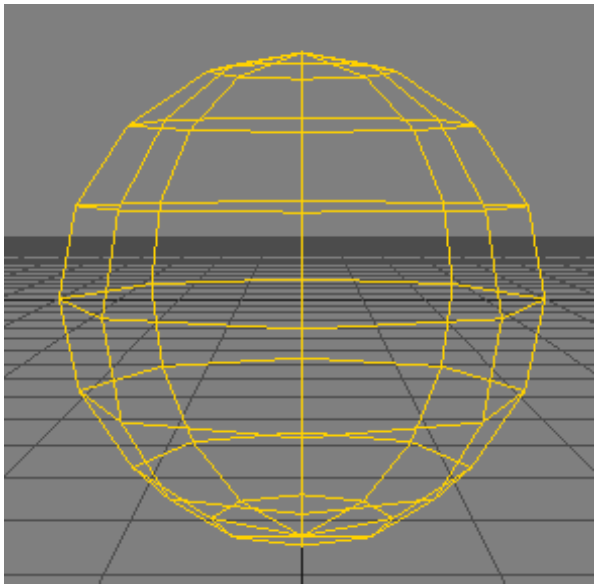
（使いたくないがデフォルトで有効・・・）

～ノード情報の取得～
まとめ

- SDKエクスプローラー
- 選択/非選択
- 名前
- ノードの種類
- 親子構造
- 位置情報

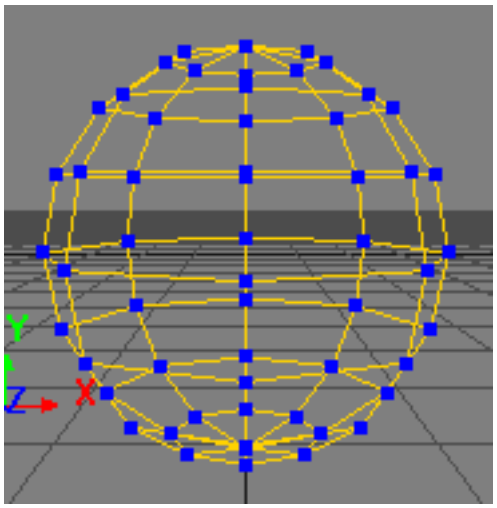
ジオメトリ情報の取得

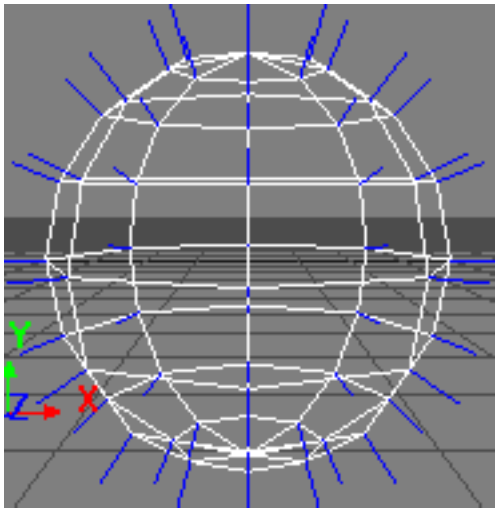
- 出力する情報
- PolygonMeshクラス
- CGeometryAccessorクラス
- 各種要素
- インデックス
- ポリゴン⇔マテリアル

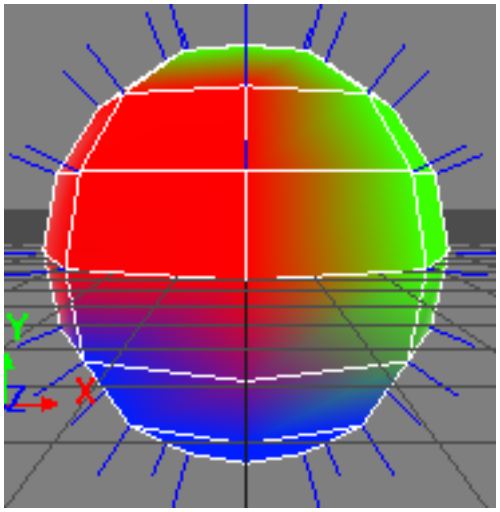


～ジオメトリ情報の取得～
出力する情報

- ポリゴンの情報を出力する
  - 各種要素（位置、法線、頂点カラー、UV、…）
  - ポリゴンのインデックス
  - ポリゴン⇔マテリアルの関係

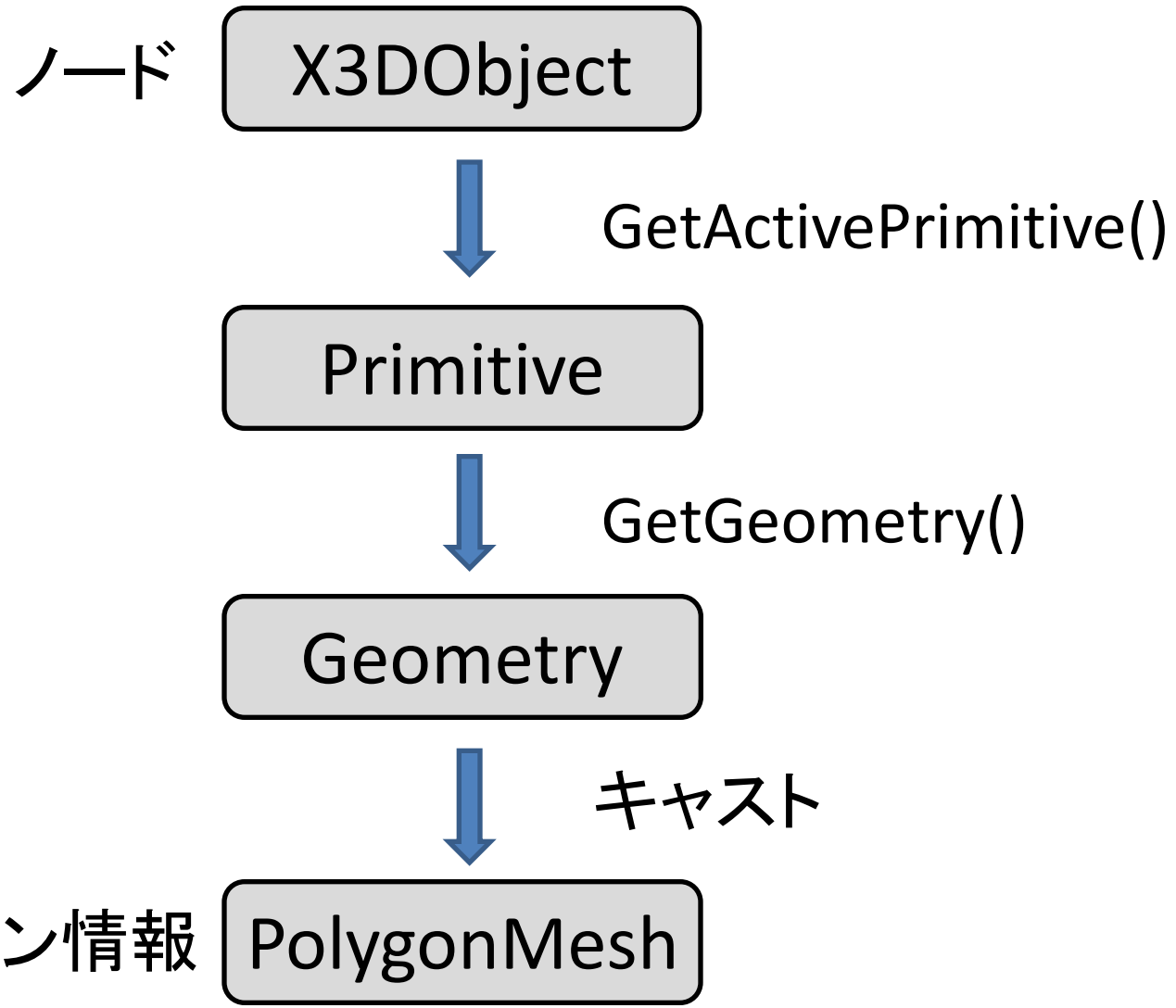






～ジオメトリ情報の取得～

X3DObject → PolygonMesh



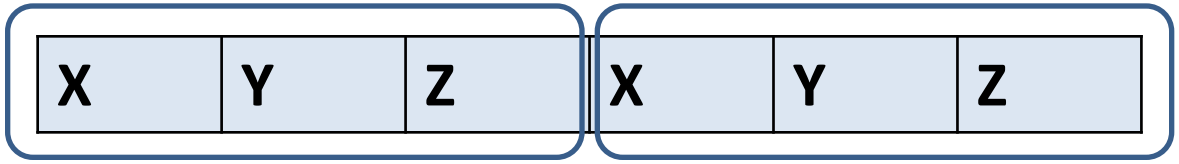
～ジオメトリ情報の取得～
CGeometryAccessorクラス

- **エクスポーター用の便利クラス**
- PolygonMesh::**GetGeometryAccessor**()

```
CGeometryAccessor ga;
ga = polyMesh.GetGeometryAccessor();
```

～ジオメトリ情報の取得～
位置情報

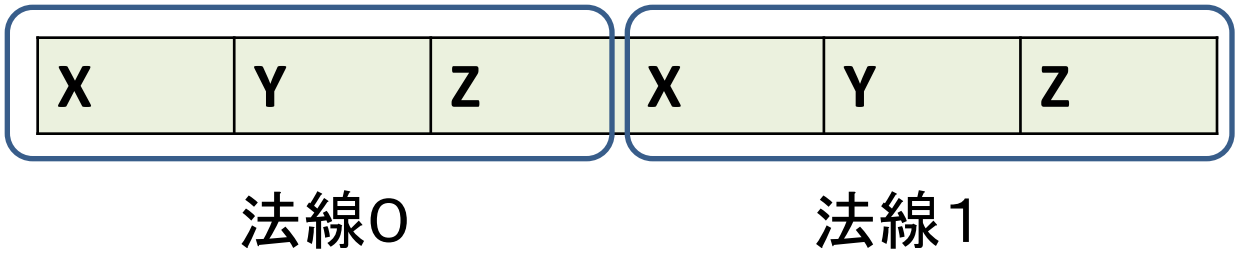
- GetVertexPositions()で取得
- データの数は位置情報x3



位置0　　位置1

～ジオメトリ情報の取得～
法線情報

- **GetNodeNormals**()で取得
- データの数は法線情報x3



- 法線情報の数 ≠ 位置情報の数

→ **別インデックスでアクセス（後述）**

～ジオメトリ情報の取得～

頂点カラー情報

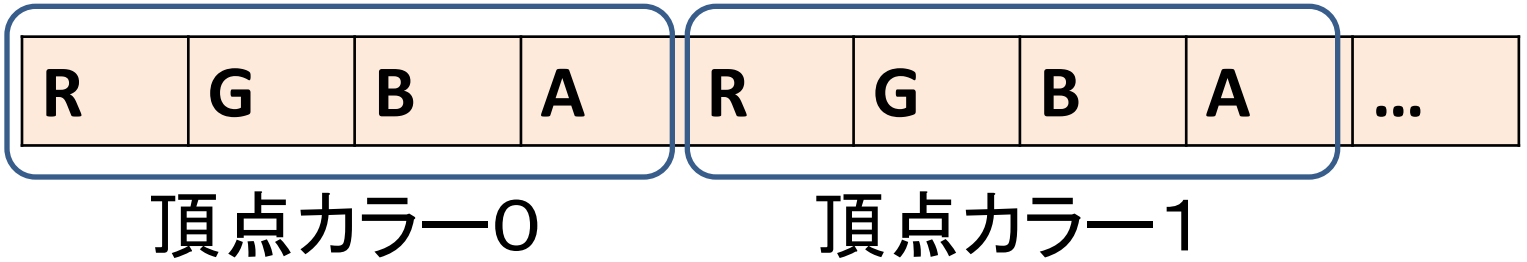
- **GetVertexColors**()で取得
- **頂点カラーは複数存在する**
- **名前**が取得できる

頂点カラーA	R	G	B	A	R	…
頂点カラーB	R	G	B	A	R	…
頂点カラーC	R	G	B	A	R	…

～ジオメトリ情報の取得～

頂点カラー情報

- 要素数は頂点カラー数x4



- 値は常に0～1とは限らない

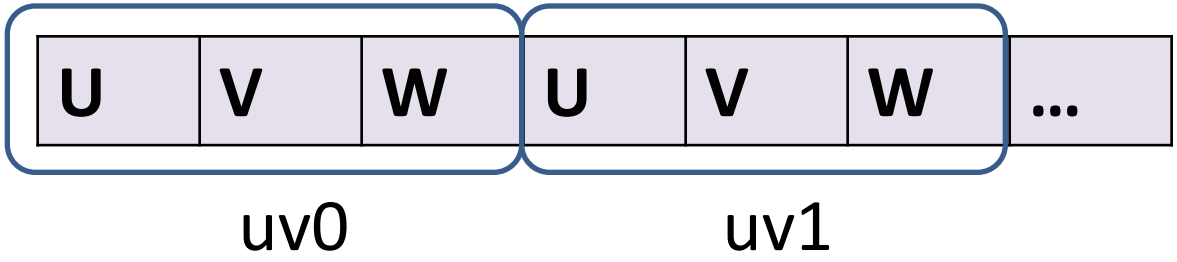
～ジオメトリ情報の取得～
UV情報

- **GetUVs**()で取得
- **UVは複数存在する**
- **名前**が取得できる

UV情報A	U	V	W	U	V	…
UV情報B	U	V	W	U	V	…
UV情報C	U	V	W	U	V	…

～ジオメトリ情報の取得～
UV情報

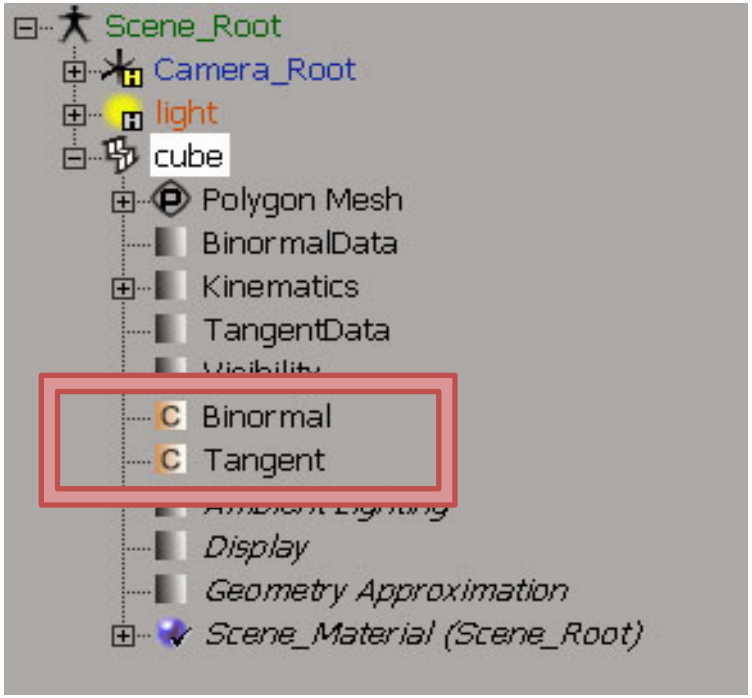
- 要素数はUVの数x3（注意！）



- W成分が不要な場合は無視

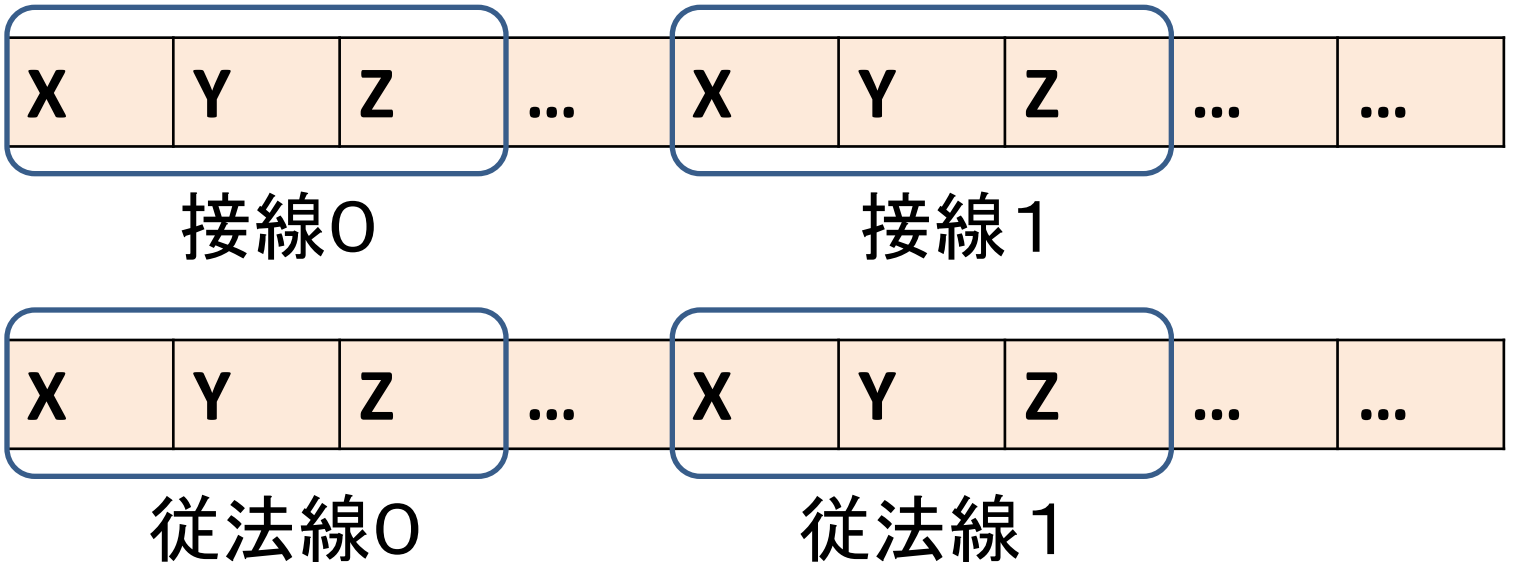
～ジオメトリ情報の取得～
接線、従法線情報

- 初期状態では接線と従法線は**存在しない**
- プロパティ＞タンジェントから追加



～ジオメトリ情報の取得～
接線、従法線情報

- 接線、従法線は複数存在する
- **頂点カラーの一種** (Softimage2010の場合)
- 値は4要素（最初の3つがベクトルに相当）



～ジオメトリ情報の取得～
インデックス情報

- 各要素の情報を取得できた

	位置
0	(0,0,10)
1	(0,10,0)
2	(10,0,0)
3	(…)

	法線
0	(0,0,1)
1	(0,1,0)
2	(1,0,0)
3	(…)

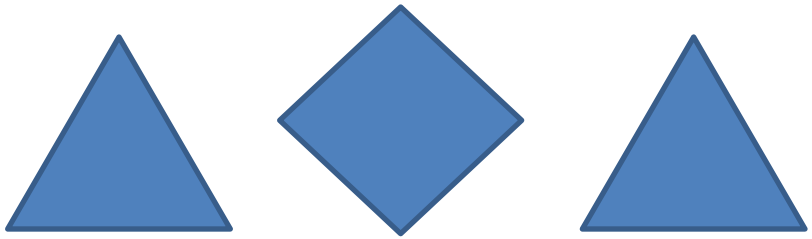
	頂点カラー
0	(1,0,1,1)
1	(0,1,0,1)
2	(1,0,0,1)
3	(…)

- 次は**ポリゴンのインデックス**

～ジオメトリ情報の取得～

インデックス情報

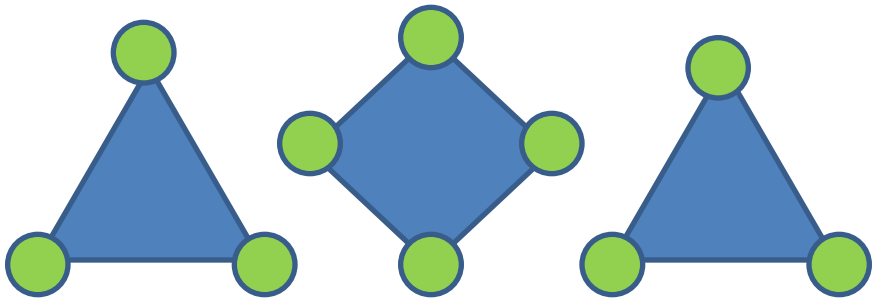
- 各ポリゴンごとの頂点数
- ポリゴンのインデックス配列
  - 頂点インデックス
  - ノードインデックス



～ジオメトリ情報の取得～

各ポリゴンの頂点数

- GetPolygonVerticesCount()から取得
- **頂点数の配列**が返る



各ポリゴンの頂点数

3	4	3

- インデックスの配列と一緒に使う

～ジオメトリ情報の取得～

頂点インデックス

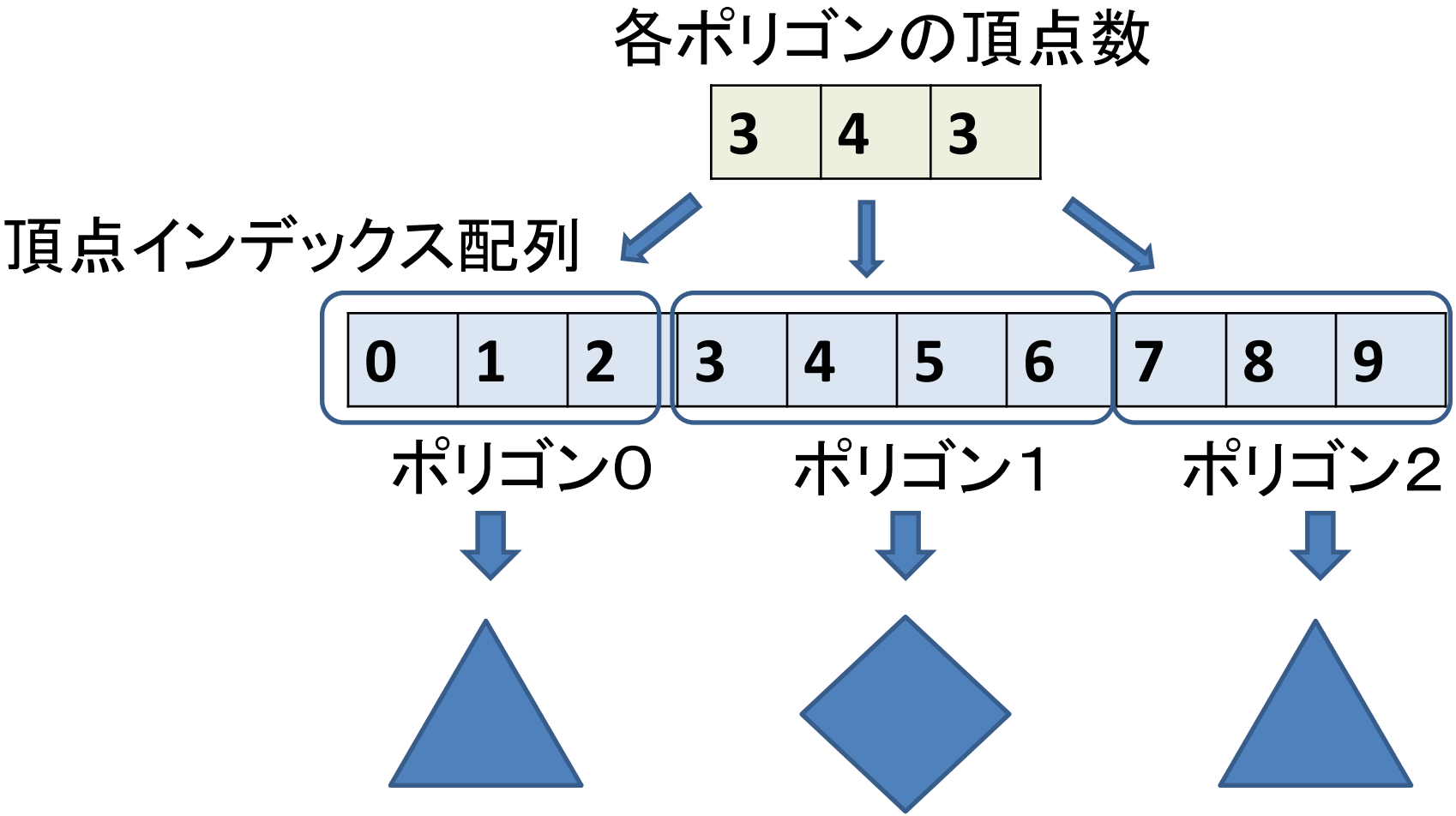
- **位置情報のためのインデックス**
- **GetVertexIndices**()から取得
- 結果は頂点インデックスの配列

0	1	2	3	4	5	6	7	8	…

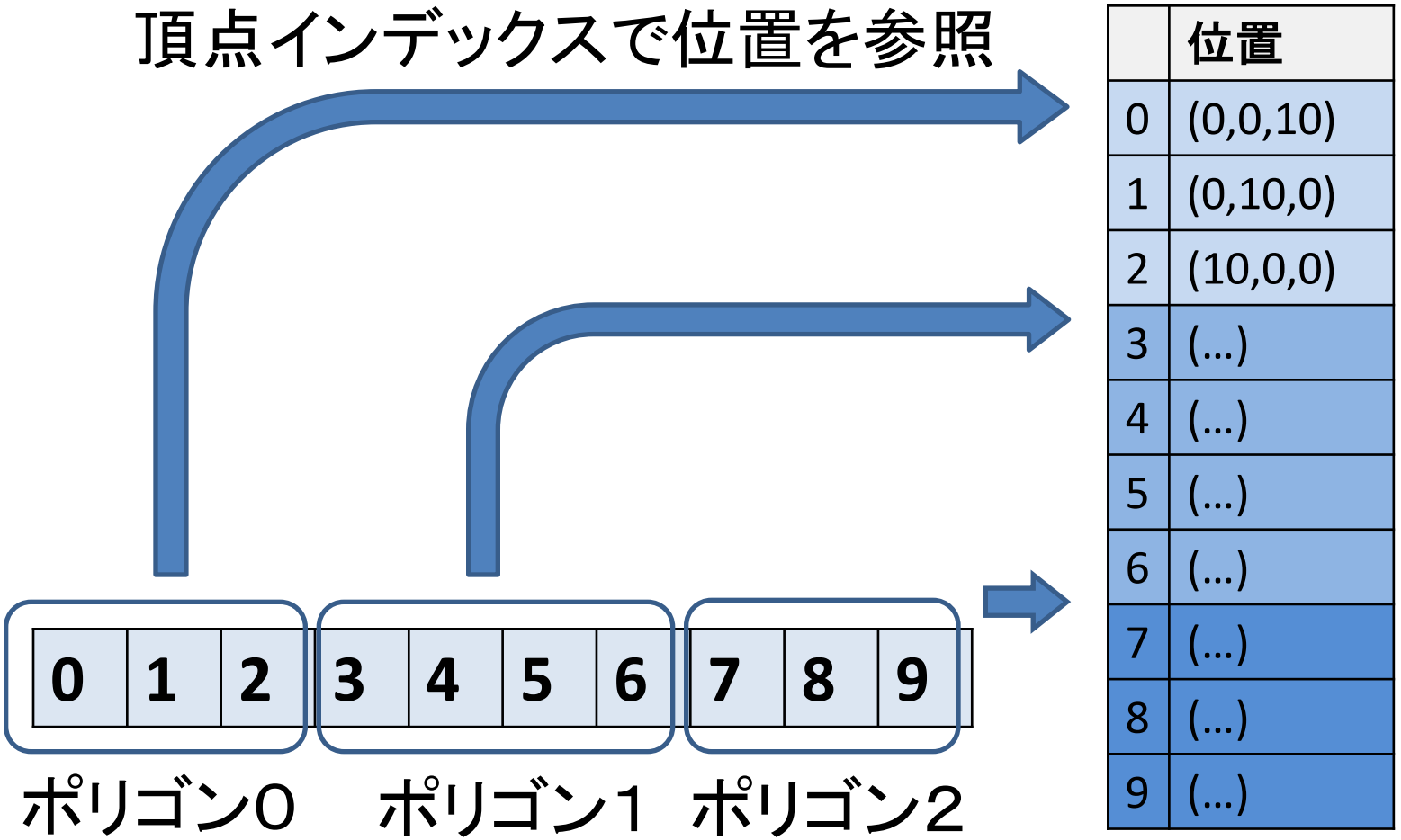
- 「各ポリゴンの頂点数」と一緒に使う

～ジオメトリ情報の取得～

頂点インデックス



～ジオメトリ情報の取得～
頂点インデックス（位置との対応）



	位置
0	(0,0,10)
1	(0,10,0)
2	(10,0,0)
3	(…)
4	(…)
5	(…)
6	(…)
7	(…)
8	(…)
9	(…)

～ジオメトリ情報の取得～

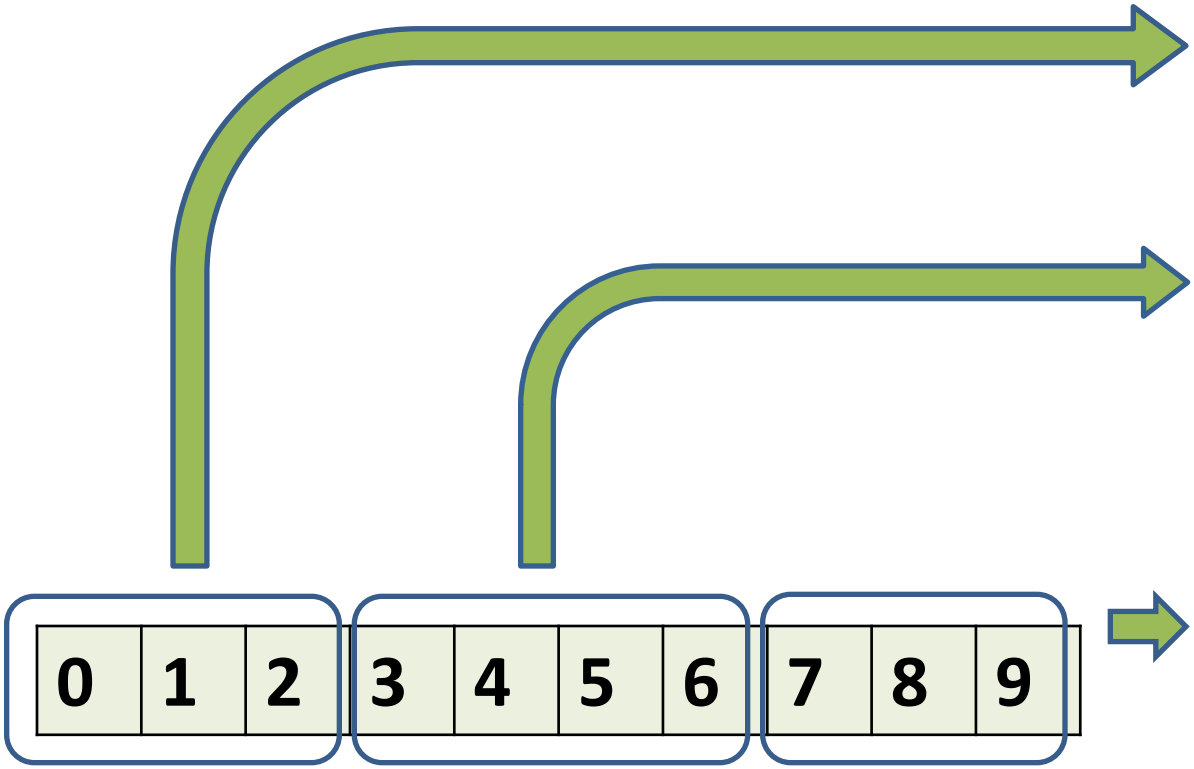
ノードインデックス

- **法線や頂点カラーのためのインデックス**
- **GetNodeIndices**()から取得
- 結果はノードインデックスの配列

0	1	2	3	4	5	6	7	8	…

～ジオメトリ情報の取得～
ノードインデックス（法線との対応）

ノードインデックスで法線を参照



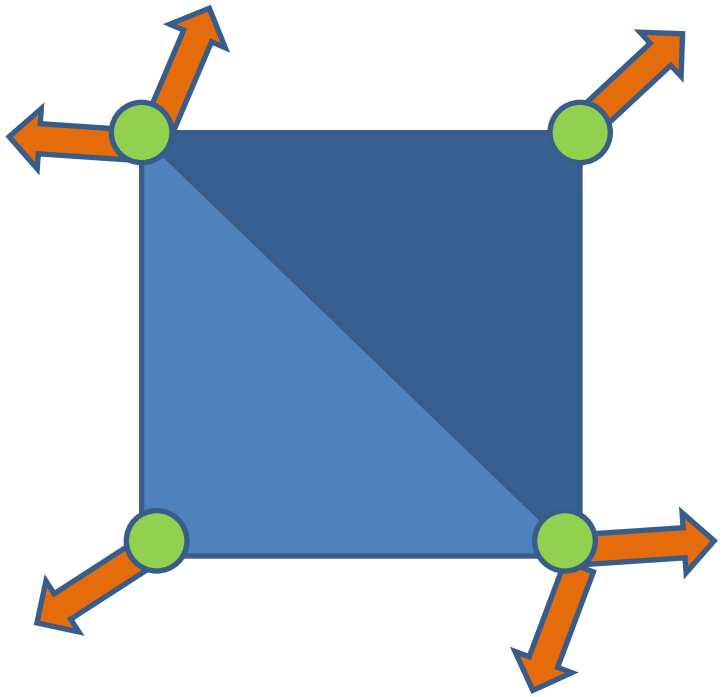
	法線
0	(0,0,1)
1	(0,1,0)
2	(1,0,0)
3	(…)
4	(…)
5	(…)
6	(…)
7	(…)
8	(…)
9	(…)

ノードインデックスの配列

～ジオメトリ情報の取得～

頂点インデックスとノードインデックス

- 何でインデックスが2つあるんだろう？
- 2つのインデックスの違いを理解しよう



3頂点x2ポリゴン

位置x4

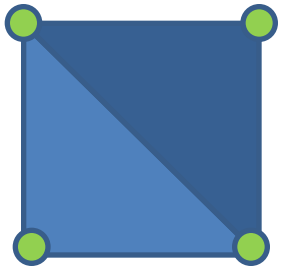
法線x6

頂点インデックスx6

ノードインデックスx6

～ジオメトリ情報の取得～

頂点とノードのインデックスの違い

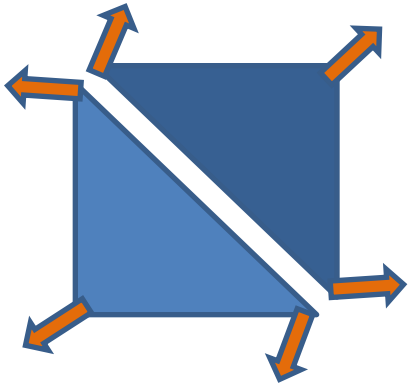


頂点インデックス

0	1	2	1	2	3

ポリゴン0　ポリゴン1

	位置
0	(…)
1	(…)
2	(…)
3	(…)



ノードインデックス

0	1	2	3	4	5

ポリゴン0　ポリゴン1

	法線
0	(…)
1	(…)
2	(…)
3	(…)
4	(…)
5	(…)

～ジオメトリ情報の取得～

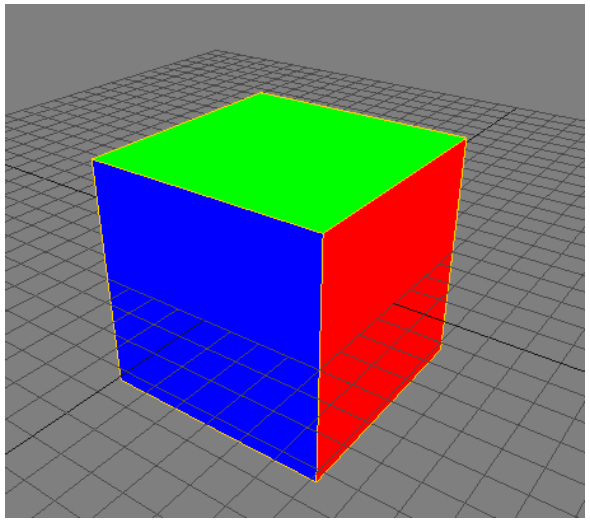
頂点とノードのインデックスの違い

- 頂点インデックスでアクセス
  - 位置
- ノードインデックスでアクセス
  - 法線
  - 頂点カラー
  - UV
  - 接線/従法線

～ジオメトリ情報の取得～

ポリゴンのマテリアル情報

- ポリゴンとマテリアルの対応関係を調べよう
- マテリアルは**ポリゴン単位**で割り当てられる



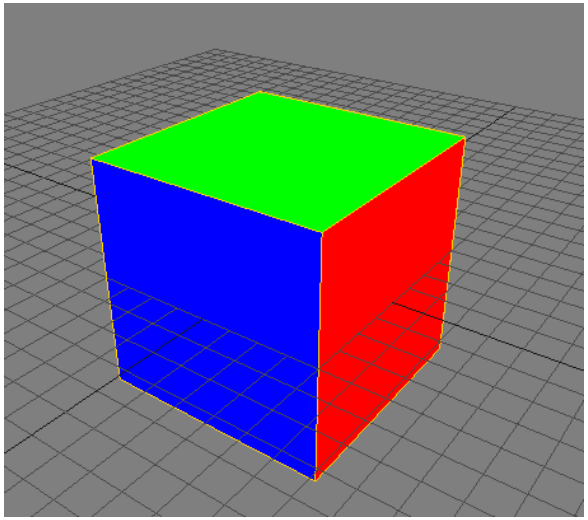
キューブ（マテリアルx3）

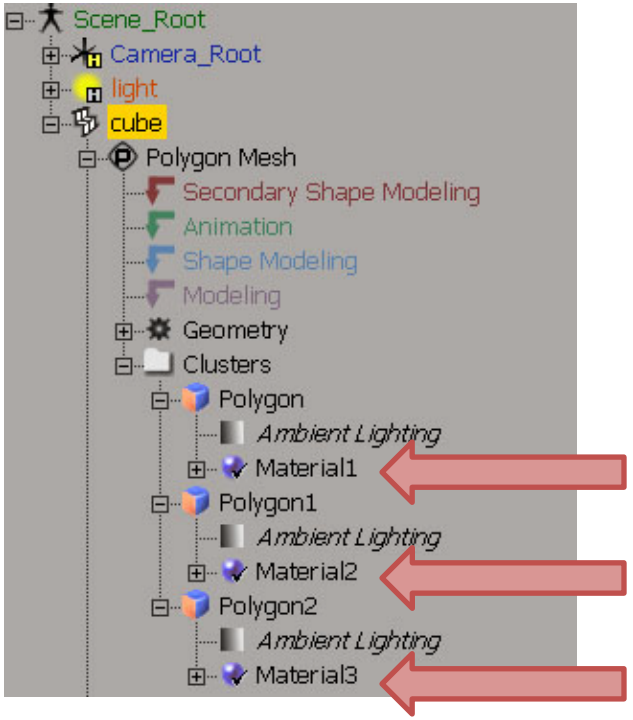
- 一つのポリゴンメッシュ

→ **一つ以上のマテリアルを使用**

～ジオメトリ情報の取得～
使用しているマテリアル

- ポリゴンメッシュが使っているマテリアルの一覧を取得





～ジオメトリ情報の取得～
使用しているマテリアル

- **GetMaterials**()
- 使っているマテリアルの一覧が返る

```
CRefArray materials = ga.GetMaterials();
Material material = materials[ 0 ];
```

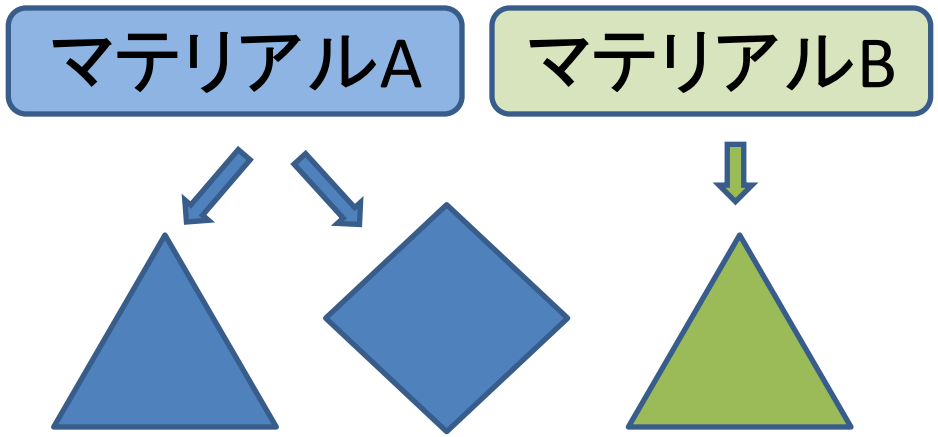
～ジオメトリ情報の取得～

ポリゴンとマテリアルの対応

- マテリアル⇔ポリゴンの対応を調べる
- **GetPolygonMaterialIndicesByMaterial**()
- 引数は**調べたいマテリアル**
- 結果は**Boolの配列**

（マテリアルを使っている場合はtrue）

～ジオメトリ情報の取得～
ポリゴンとマテリアルの対応



- GetPolygonMaterialIndicesByMaterial()

マテリアルA	**true**	**true**	**false**
マテリアルB	**false**	**false**	**true**

～ジオメトリ情報の取得～
まとめ

- 出力する情報
- PolygonMeshクラス
- CGeometryAccessorクラス
- 各種要素
- インデックス
- ポリゴン⇔マテリアル

マテリアル情報の取得

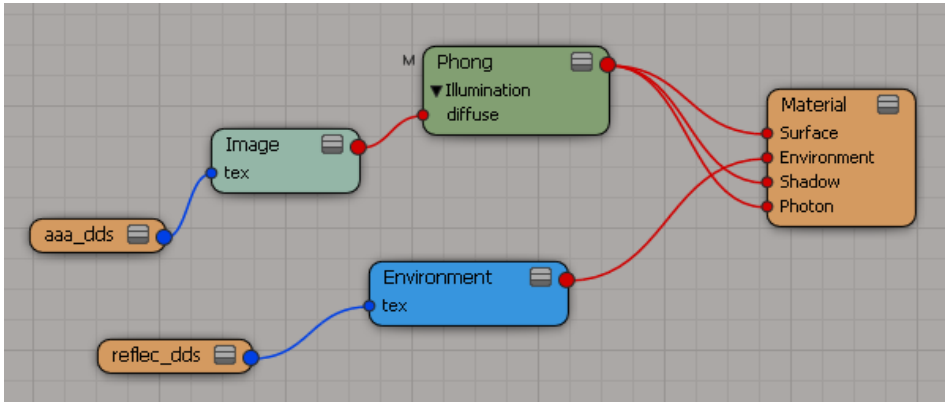
- 質感情報の設定方法
- レンダーツリーからの情報取得
  - テクスチャ
  - パラメータ

～マテリアル情報の取得～
質感設定の方法

- 「ゲーム用」の質感情報を設定
  - 外部エンジン/ツール
  - DCCツール
    - ユーザー定義パラメータ
    - レンダーツリー
    - リアルタイムシェーダー

～マテリアル情報の取得～
レンダーツリー

- Softimageのレンダーツリーとは？
- **ツリー構造**で質感設定を行う

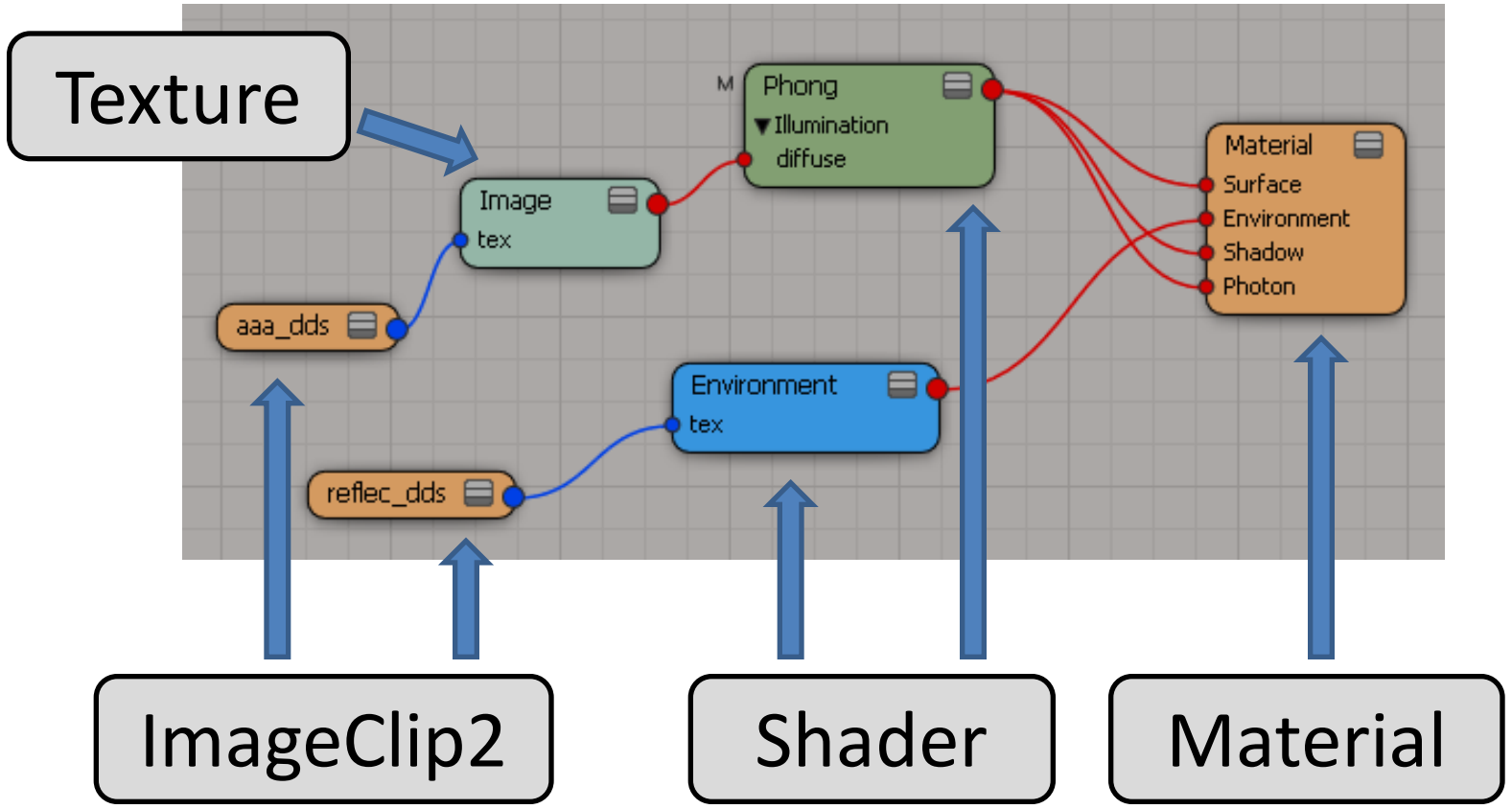


→ ツリー構造から必要な情報を取得

- テクスチャ
- 各種パラメータ

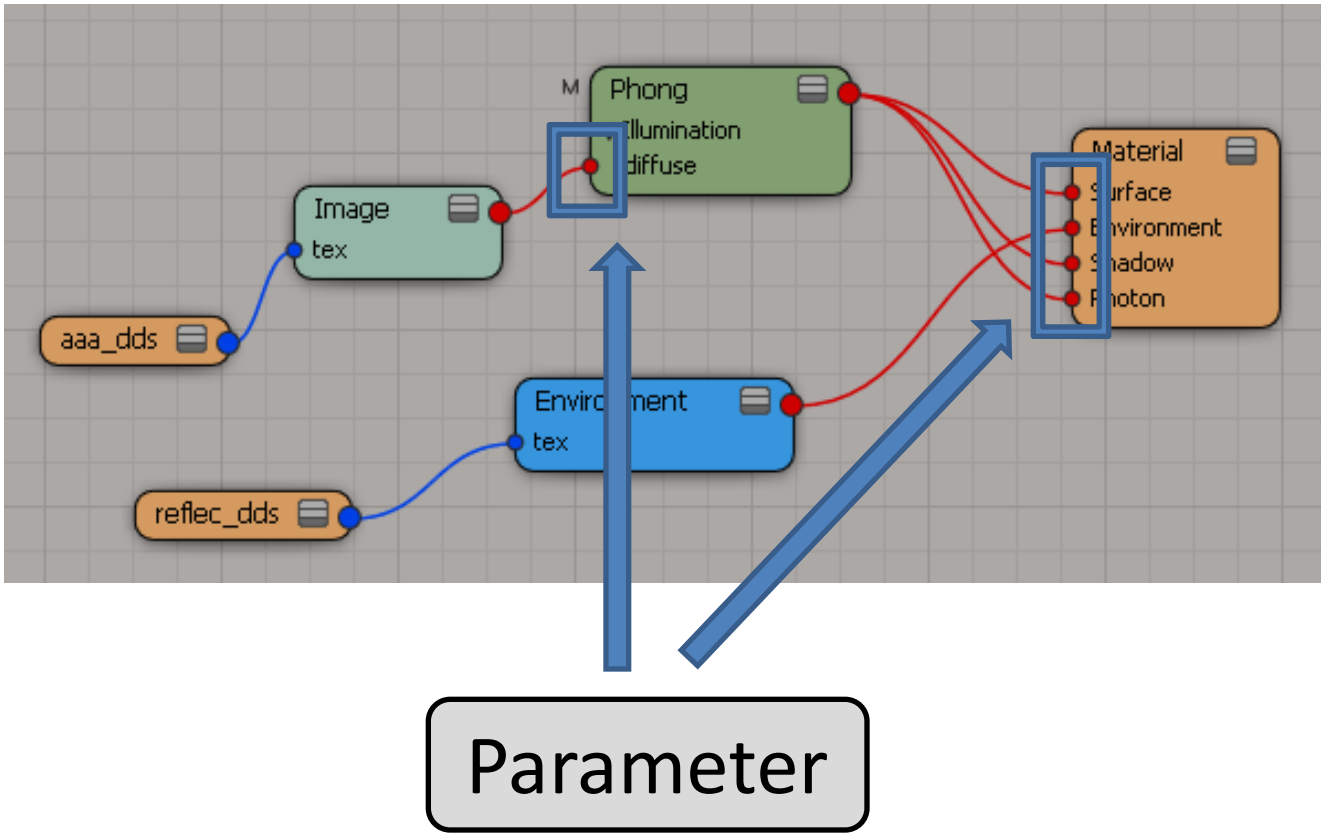
～マテリアル情報の取得～

レンダーツリーとSDKクラスの対応



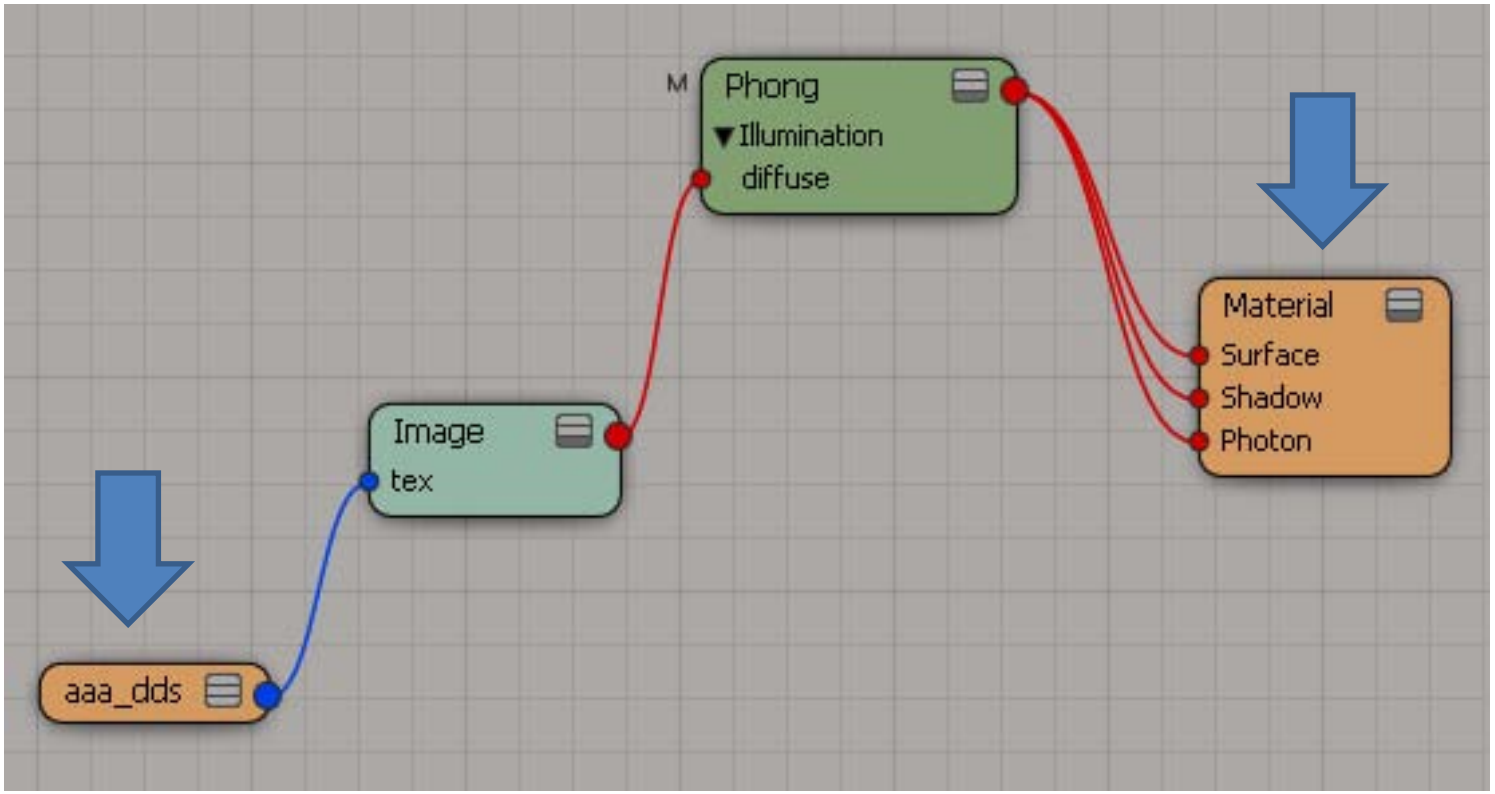
～マテリアル情報の取得～

レンダーツリーとクラスの対応



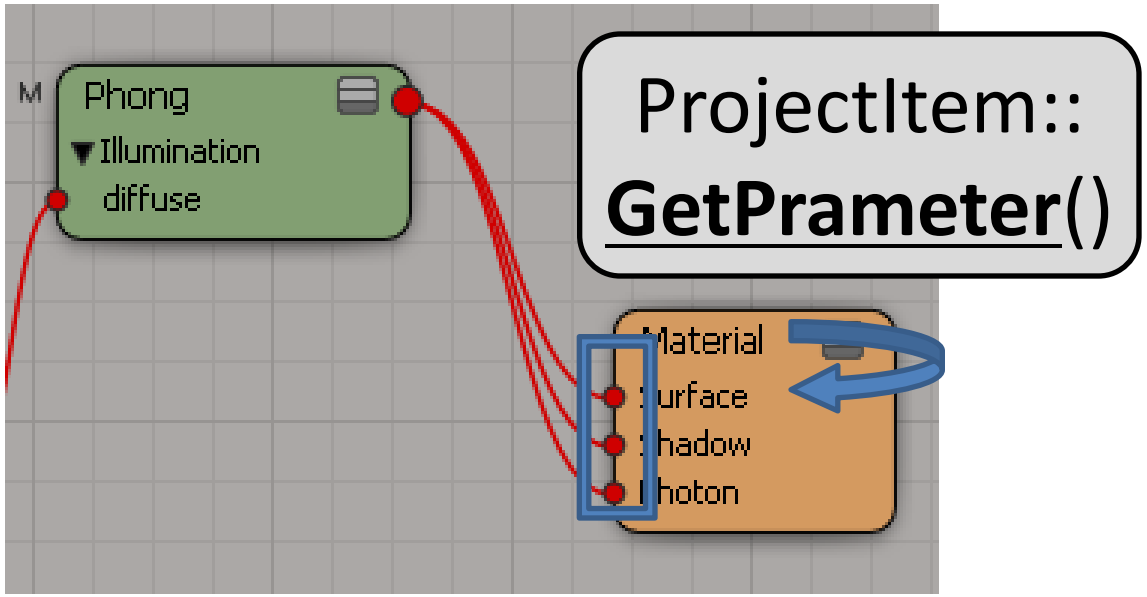
～マテリアル情報の取得～
マテリアル　→　イメージクリップ

- 接続をたどって各ノードを取得しよう



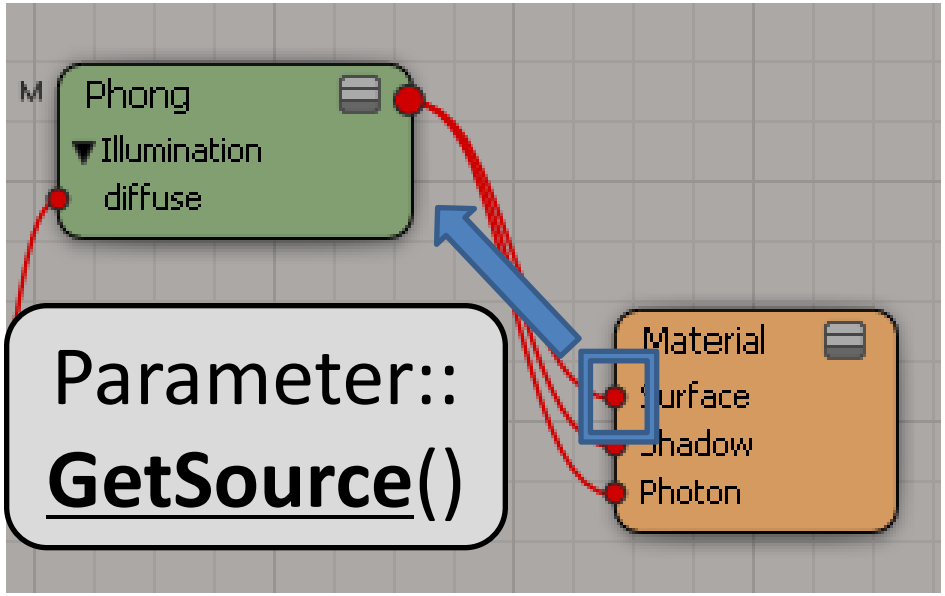
～マテリアル情報の取得～
マテリアル　→　パラメータ

- マテリアルのパラメータ
- ProjectItem::**GetParameter**()



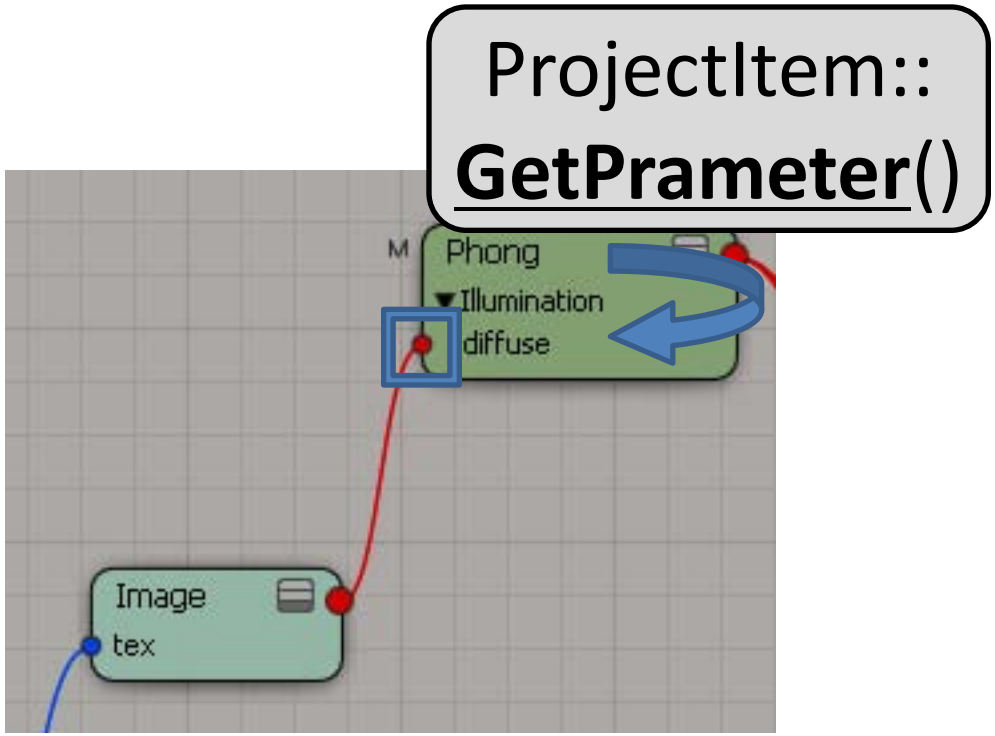
～マテリアル情報の取得～
パラメータの接続元

- パラメータ　→　シェーダーノード
- Parameter::**GetSource**()
- 未接続の場合は無効なオブジェクト



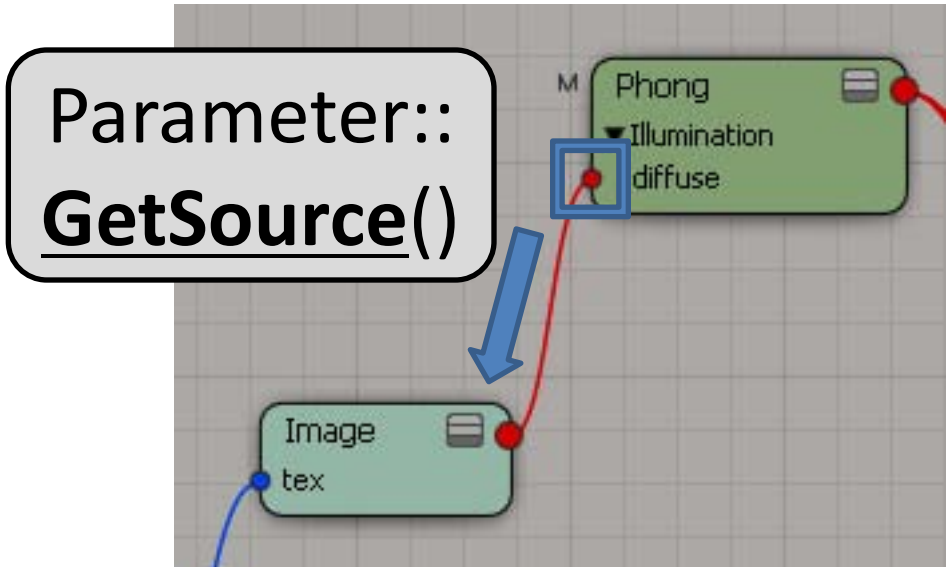
～マテリアル情報の取得～
シェーダーノード → パラメータ

- シェーダーノードのパラメータ
- ProjectItem::**GetParameter**()



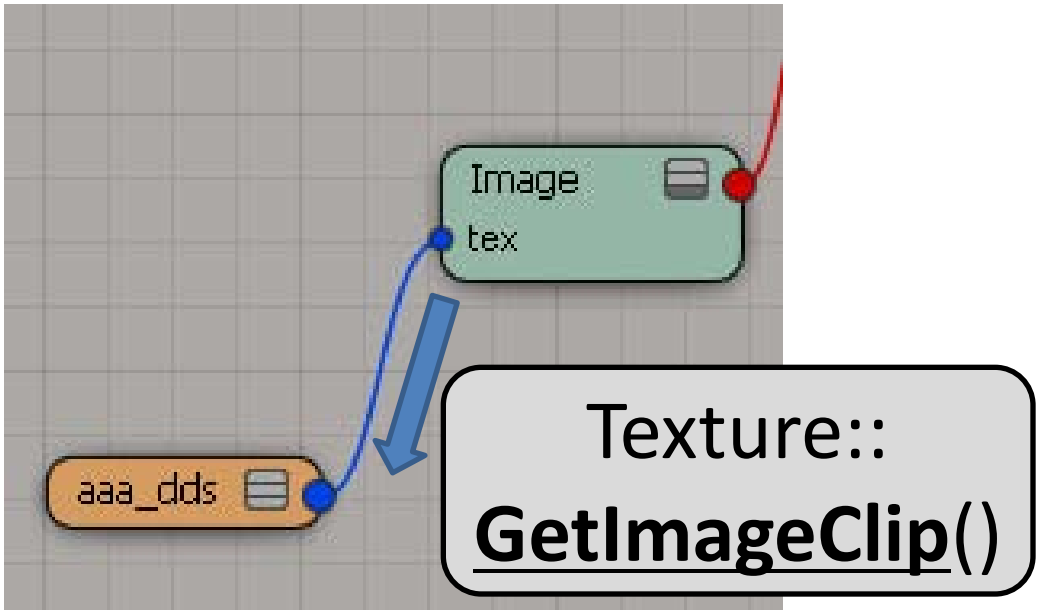
～マテリアル情報の取得～
パラメータの接続元

- パラメータの接続元は**GetSource**()で
- 接続元はTextureクラス

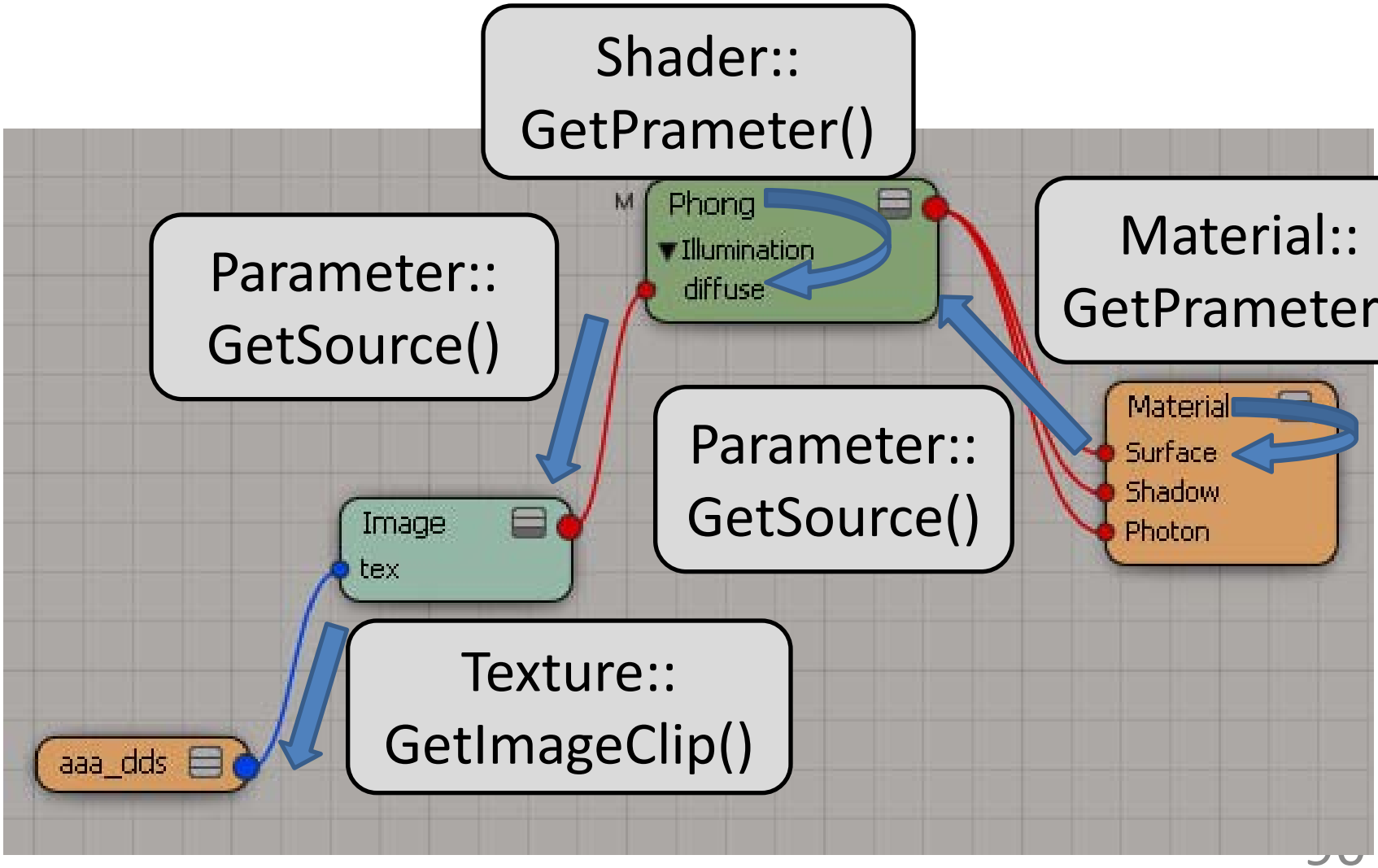


～マテリアル情報の取得～
ImageClipの取得

- テクスチャノード　→　イメージクリップ
- Texture::**GetImageClip**()



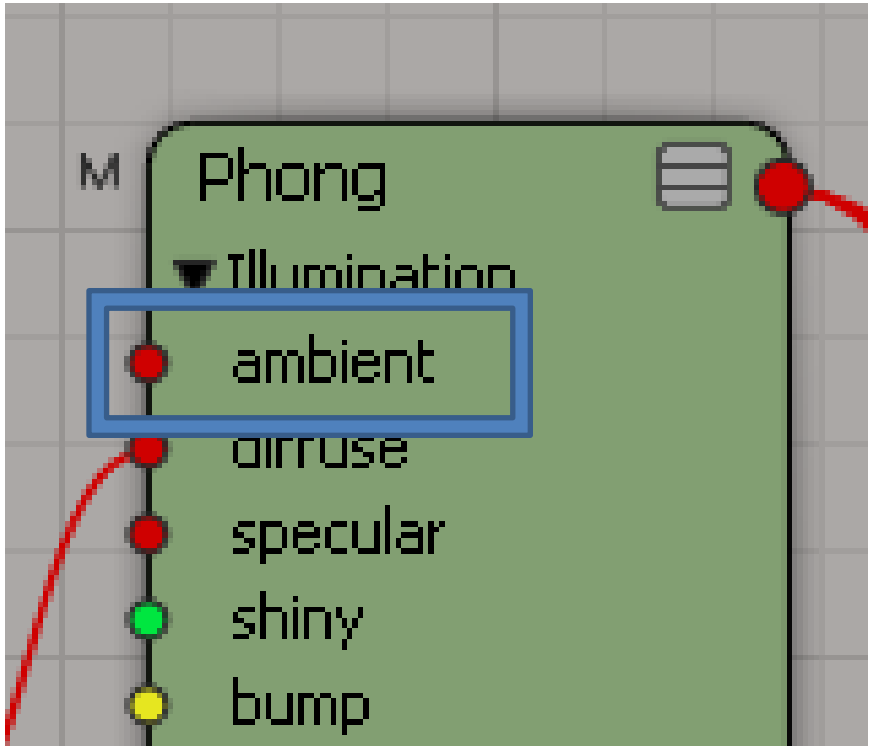
～マテリアル情報の取得～
レンダーツリーの解析

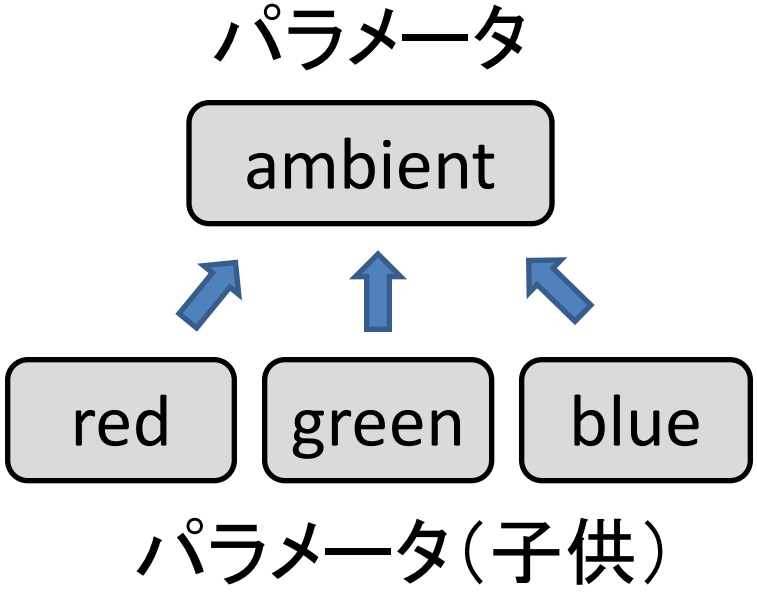


～マテリアル情報の取得～

シェーダーノードのパラメータ

- パラメータの値（カラー）を取得
- カラーのパラメータは入れ子になっている





～マテリアル情報の取得～

パラメータの値の取得

1. 親パラメータを取得
2. 子供パラメータの値を取得

ProjectItem::**GetParameterValue()**

```
Parameter amb;
amb = shader.GetParameter( L”ambient” );
…
float r = amb.GetParameterValue( L”red” );
float g = amb.GetParameterValue( L”green” );
float b = diff.GetParameterValue( L”blue” );
```

～マテリアル情報の取得～

テクスチャの情報

- 画像ファイル
- UV座標

～マテリアル情報の取得～

テクスチャの画像ファイル

- 画像ファイルの情報は**ImageClip2クラス**
- ImageClip2::**GetFileName**()

```
ImageClip2 imageClip;
…
CString filename = imgClip.GetFileName();
```

～マテリアル情報の取得～
テクスチャのUV座標

- テクスチャが使用している**UV座標の名前**
- Textureクラスの**”tspace_id”パラメータ**

```
Parameter uv;
uv = texture.GetParameter( “tspace_id” );
CString uvName = uv.GetValue();
```

- 本当はもっと複雑（環境マップの場合など）

→ ドキュメントとソースを参照

～マテリアル情報の取得～

まとめ

- 質感情報の設定方法
- レンダーツリーからの情報取得
  - テクスチャ
  - パラメータ

アニメーション情報の取得

- アニメーションの種類
- アニメーションの出力範囲
- Transformのアニメーション
- Parameterのアニメーション

～アニメーション情報の取得～

アニメーションの種類

- **ファンクションカーブ有**
  - 手付けモーション
  - モーションキャプチャ
- **ファンクションカーブ無**
  - エクスプレッション
  - コンストレイン
  - 物理シミュレーション
  - その他

～アニメーション情報の取得～

ファンクションカーブの有無

- ファンクションカーブをそのまま出力

  → すべてのアニメを出力できない

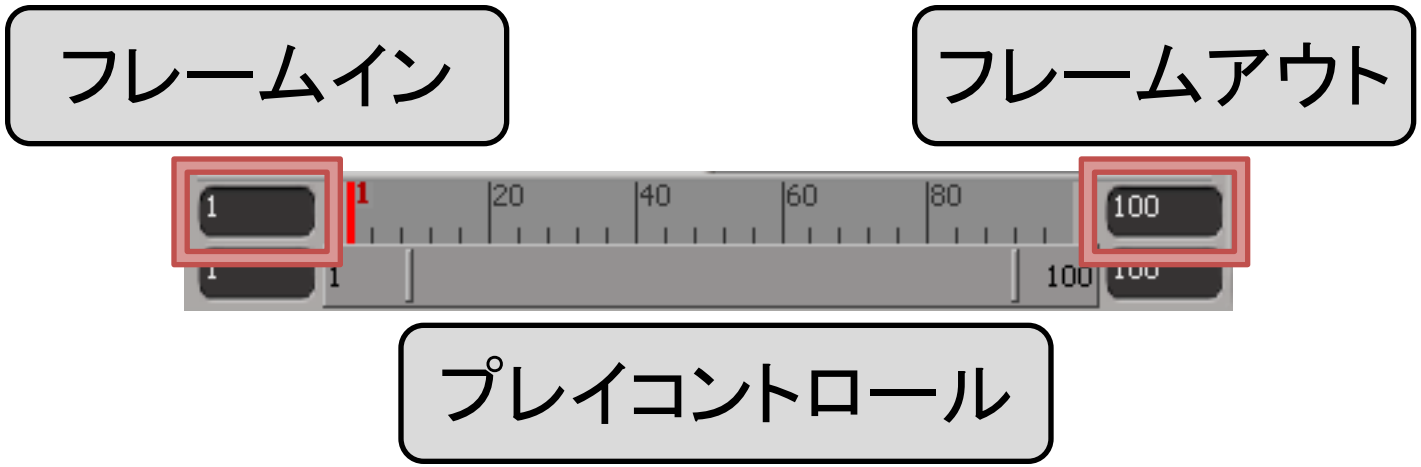
- **すべてのフレームの情報を出力**

  → すべてのアニメを出力できる

～アニメーション情報の取得～

アニメーションの出力範囲

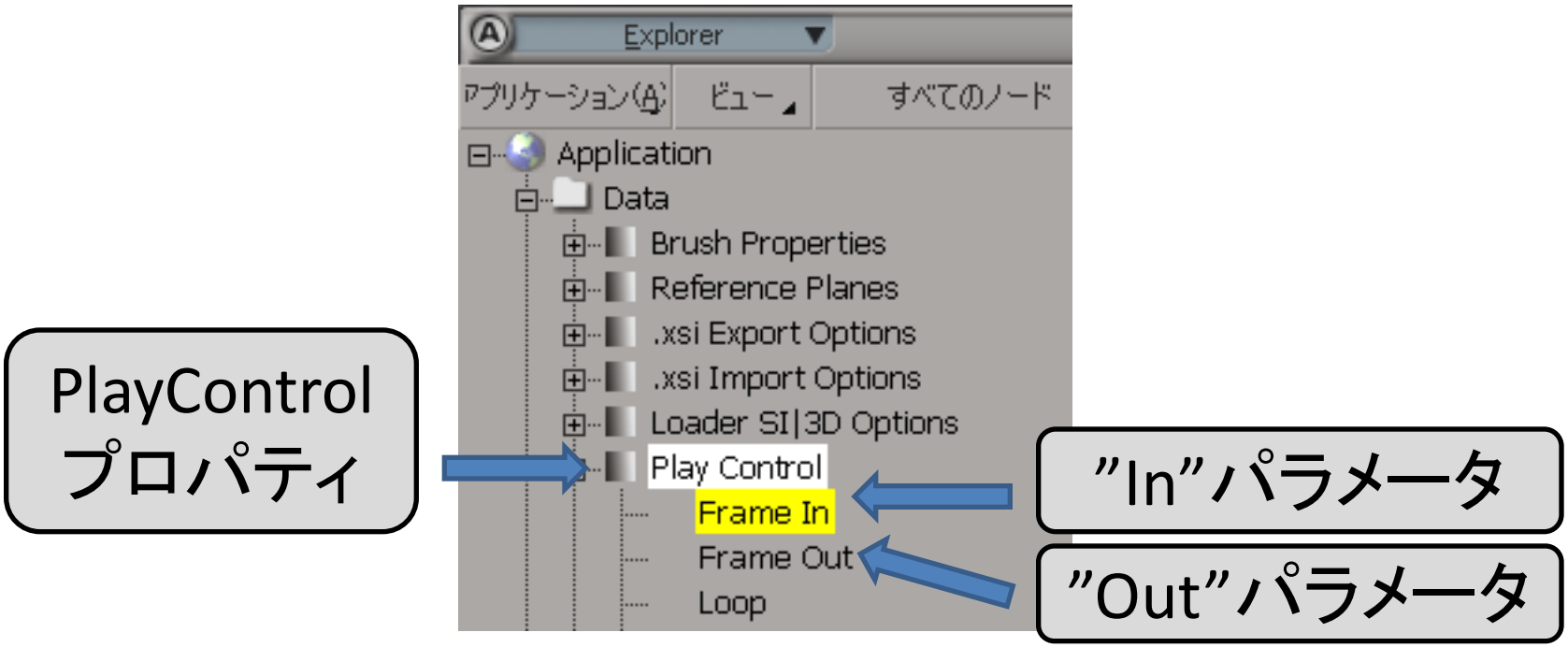
- 何フレームから何フレームまで出力する？
- プレイコントロールのフレームイン、フレームアウトの範囲を出力



～アニメーション情報の取得～

アニメーション範囲の取得

- **”Play Control”プロパティ**
- **”In”, ”Out”パラメータ**から取得



～アニメーション情報の取得～
Transformアニメーション

- KinematicState::**GetTransform**()
- **引数でフレーム数を指定する**

```
for( int frame=in; frame<=out; frame++ ) {
    CTransform trans;
    trans = kineState.GetTransform( frame );
    …
}
```

～アニメーション情報の取得～
Parameterアニメーション

- Parameter::**GetValue**()
- **引数でフレーム数を指定**

```
for( int frame=in; frame<=out; frame++ ) {
    float v = param.GetValue( frame );
    …
}
```

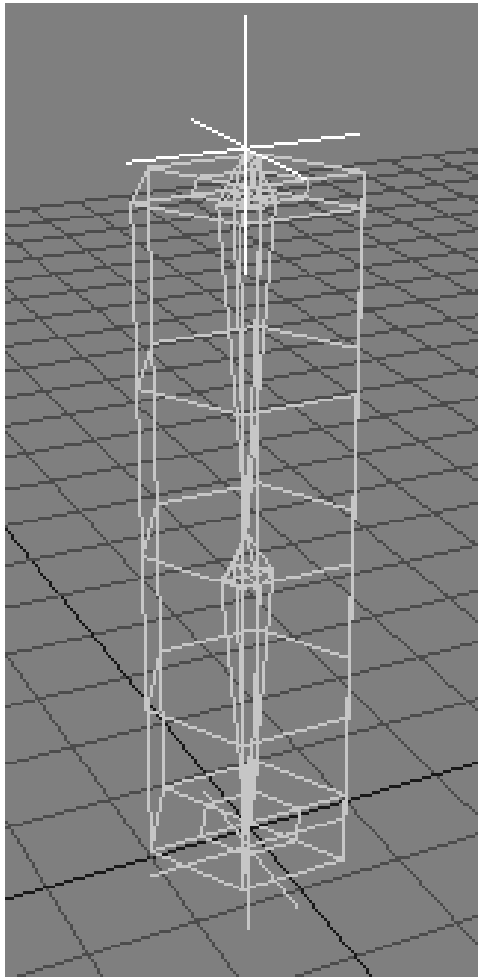
～アニメーション情報の取得～

まとめ

- アニメーションの種類
- アニメーションの出力範囲
- Transformのアニメーション
- Parameterのアニメーション

スキニング情報の取得

- 用語説明
- 出力する情報
- テストデータ
- スキニング情報の取得
  - 骨
  - ウェイト値
  - バインド時のマトリックス



～スキニング情報の取得～
用語説明

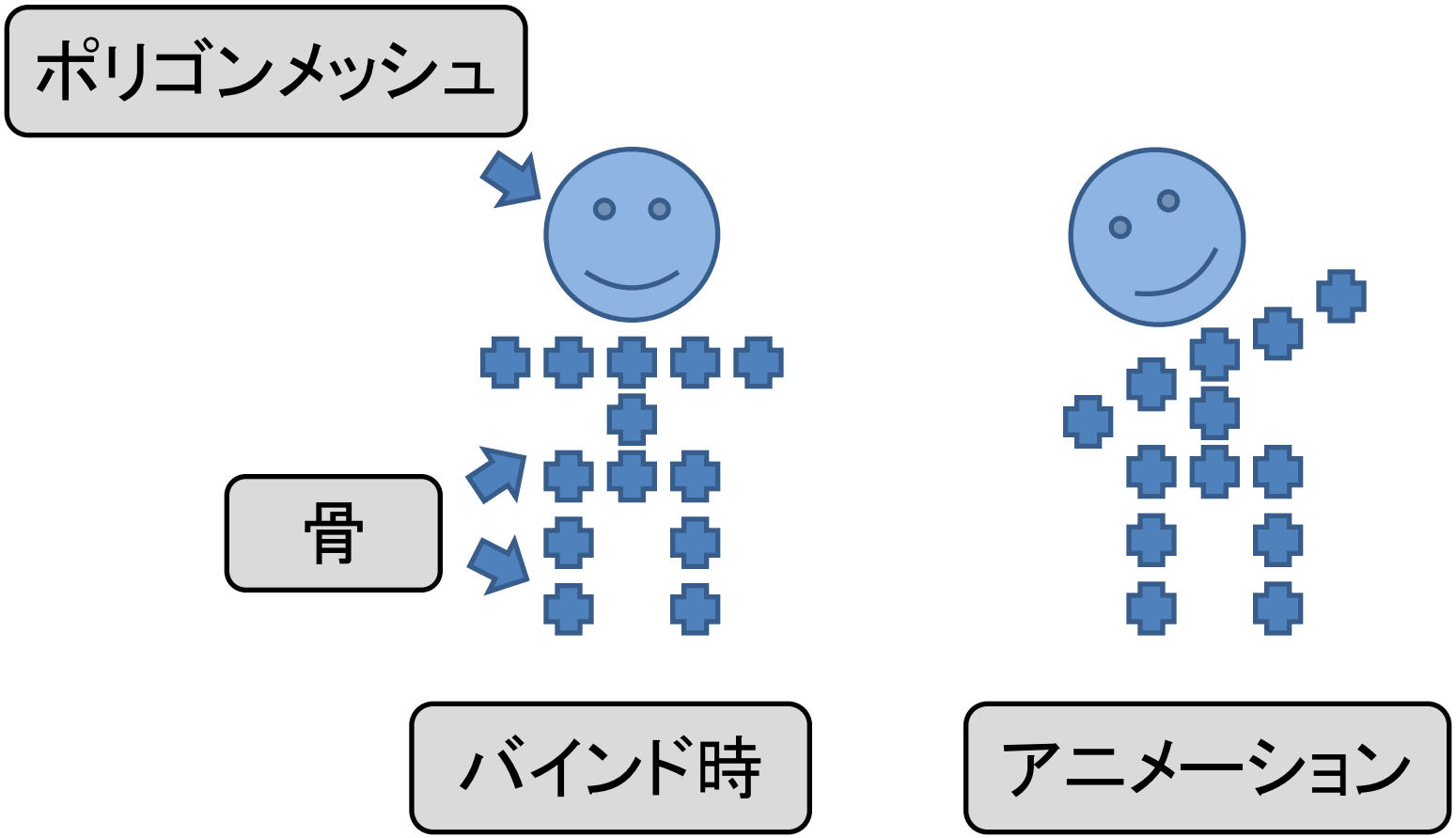
- Softimageではスキニングのことを「**エンベロープ**」と呼ぶ
- スキニングを設定することを「**バインドする**」という

～スキニング情報の取得～
出力する情報は？

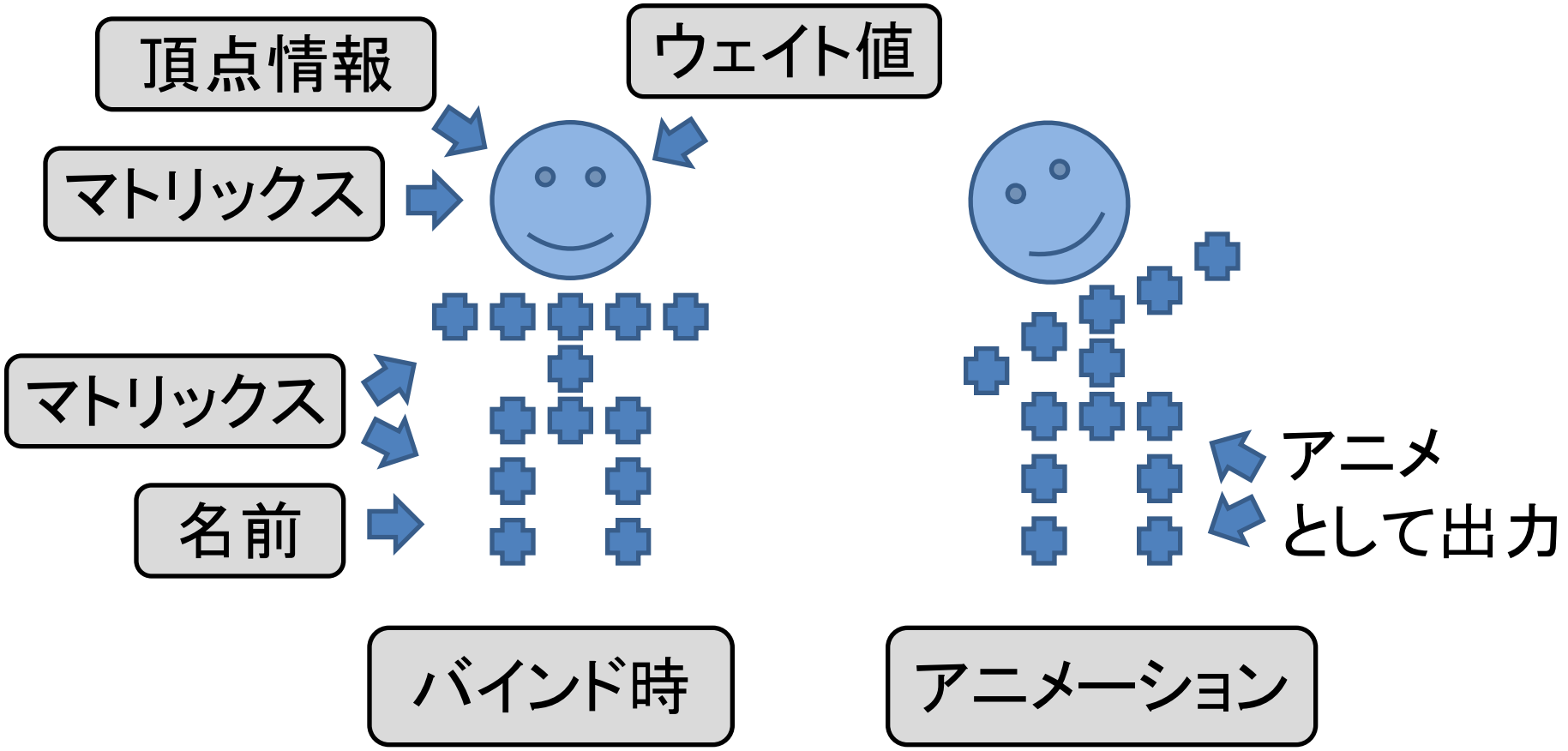
- 何を出力すればいいのだろう？

→ 詳細はドキュメントを参照

～スキニング情報の取得～
出力する情報



～スキニング情報の取得～
出力する情報



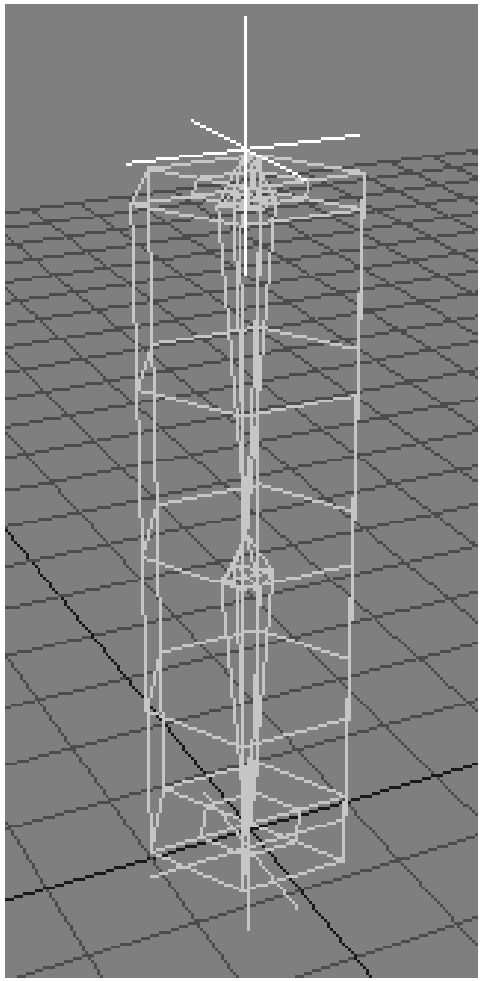
～スキニング情報の取得～
テストデータ

- テストデータは**極力シンプル**に
- ダメな例
- よい例

～スキニング情報の取得～

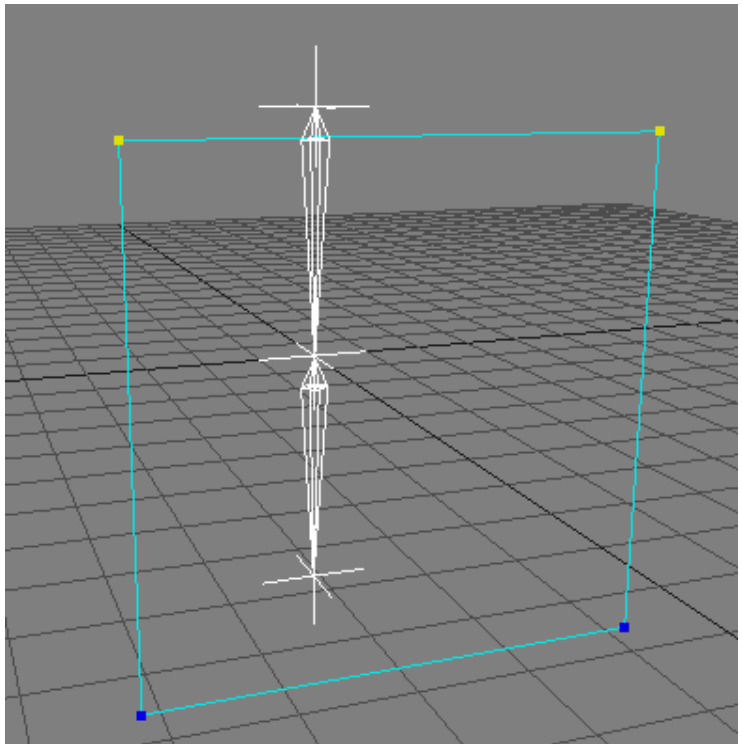
テストデータ　ダメな例

- 頂点数が多すぎる
- 頂点数：6x5＝30
- 骨：3
- **ウェイト：30x3＝90**



～スキニング情報の取得～
テストデータ　よい例

- ４角形ポリゴン1つ
- 頂点数:４
- 骨:２
- **ウェイト:4x2＝8**



～スキニング情報の取得～
Envelopeの取得

- スキニング情報は**Envelopeクラス**に格納されている
- X3DObject::**GetEnvelopes**()

```
CRefArray envelopes = x3DObject.GetEnvelopes();
```

～スキニング情報の取得～
骨の取得

- Envelope::**GetDeformers**()
- 戻り値は骨の配列

```
CRefArray deformers = envelope.GetDeformers();
```

～スキニング情報の取得～

ウェイト値の取得

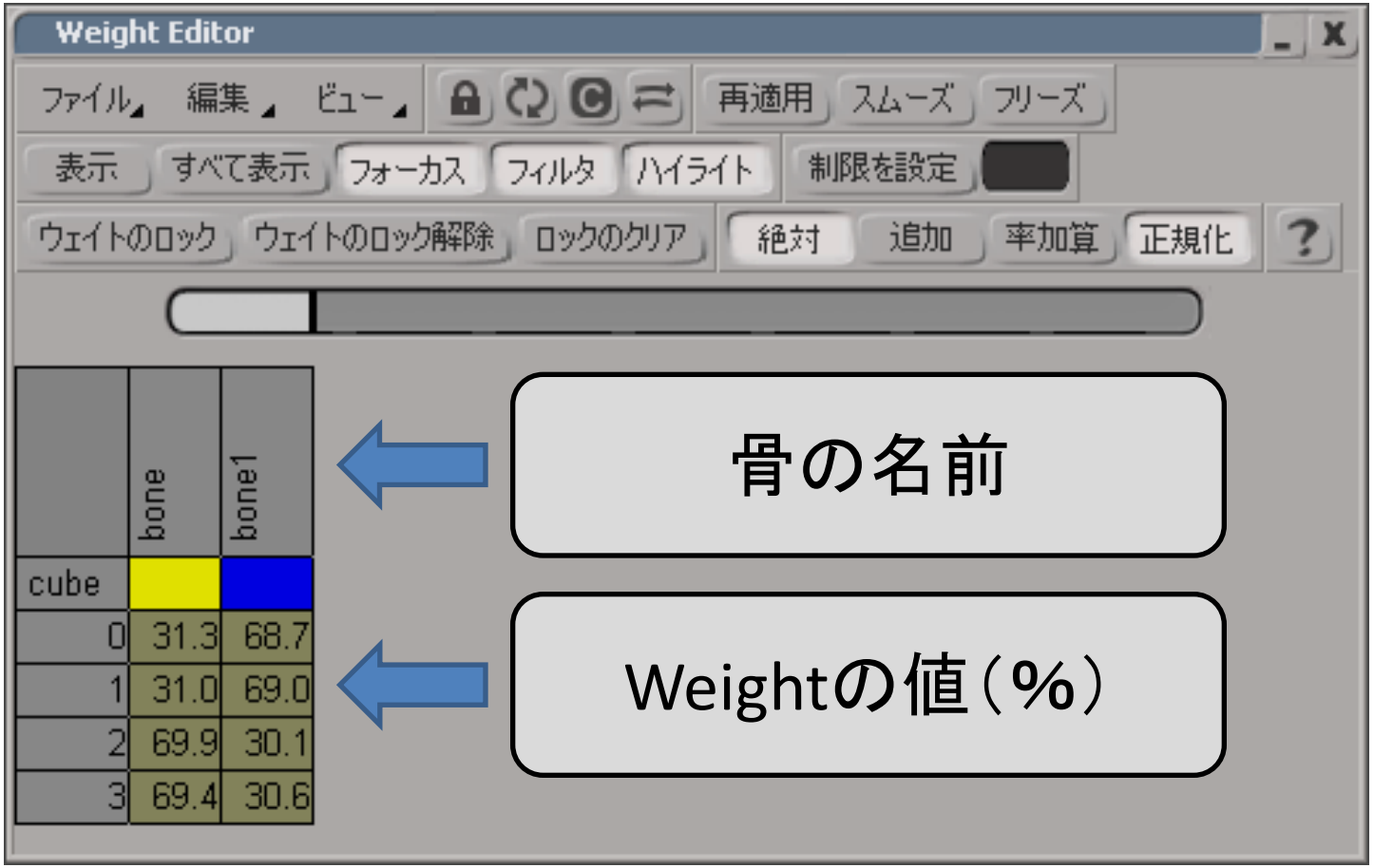
- Envelope::**GetDeformerWeights**()
- **引数に骨を指定**
- 結果は**ウェイト値（%）の配列**
- 位置の数と等しい

bone	**31.3**	**31.0**	**69.9**	**69.4**
bone1	**68.7**	**69.0**	**30.1**	**30.6**

４頂点　骨２　のウェイト値

～スキニング情報の取得～
確認

- ウェイトエディターでウェイト情報を確認



～スキニング情報の取得～
バインド時の座標

- 骨、ポリゴンメッシュの**バインド時の座標**
- X3DObject::**GetStaticKinematicState**()

```
CTransformation trs;
trs = x3DObject.GetStaticKinematicState().GetTransform();
```

～スキニング情報の取得～
まとめ

- 用語説明
- 出力する情報
- テストデータ
- スキニング情報の取得
  - 骨の名前
  - ウェイト値
  - バインド時のマトリックス

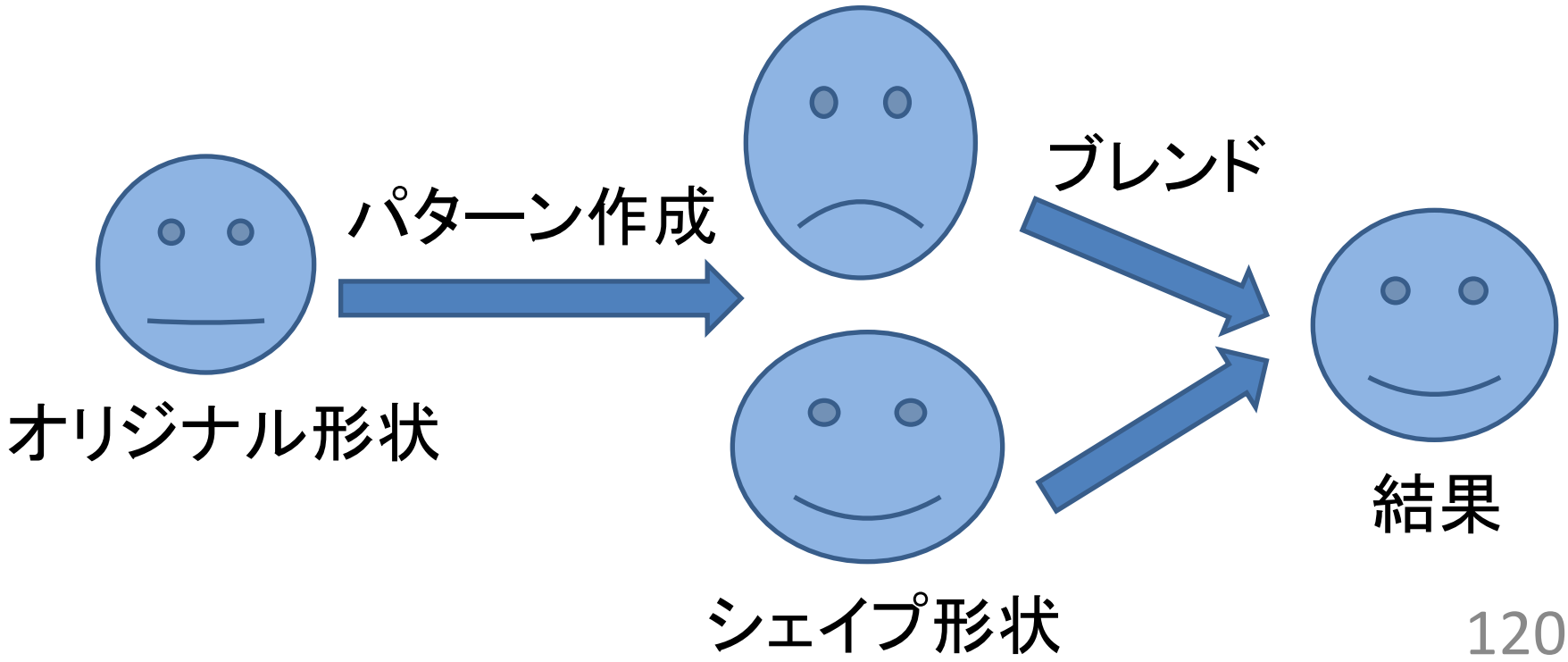
シェイプ情報の取得

- シェイプとは？
- 出力する情報
- シェイプマネージャー
- シェイプの取得方法
  - 名前
  - 形状
  - スキニング＋シェイプの場合

シェイプ情報の取得

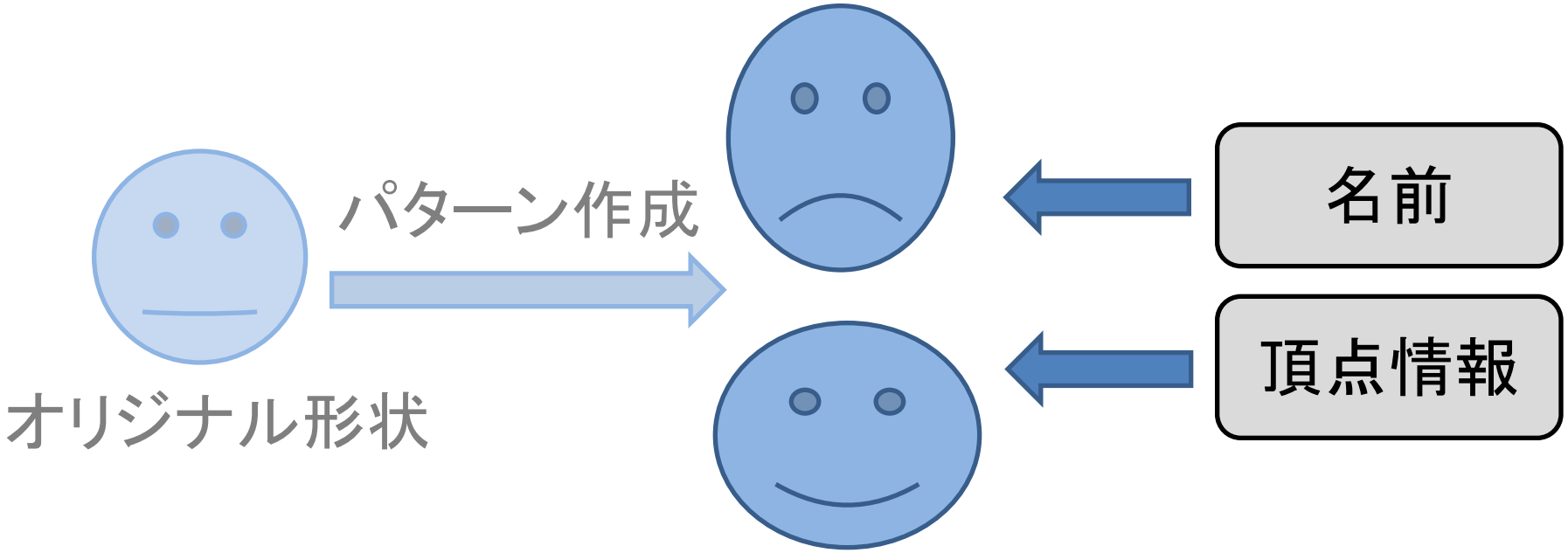
- Softimageのシェイプとは？

  → シェイプアニメーション、モーフィング



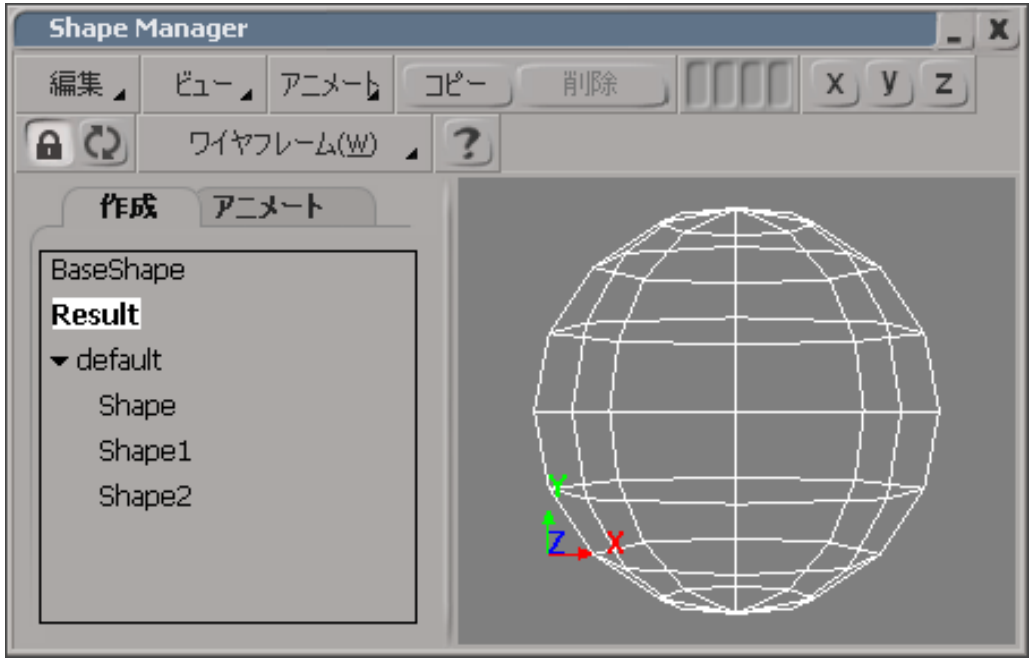
出力する情報

- シェイプの**名前**
- シェイプの**頂点情報**



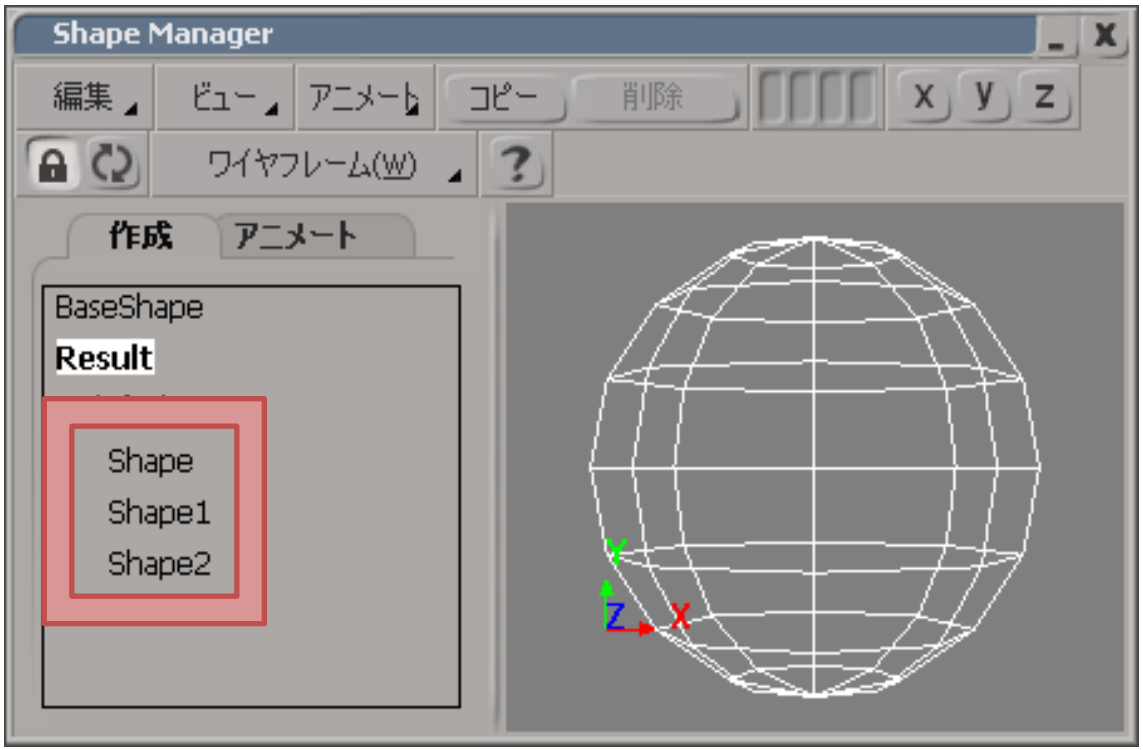
～シェイプ情報の取得～
シェイプマネージャー

- **シェイプの管理・編集ツール**
- 名前・形状の確認
- テストデータの作成



～シェイプ情報の取得～
シェイプの名前

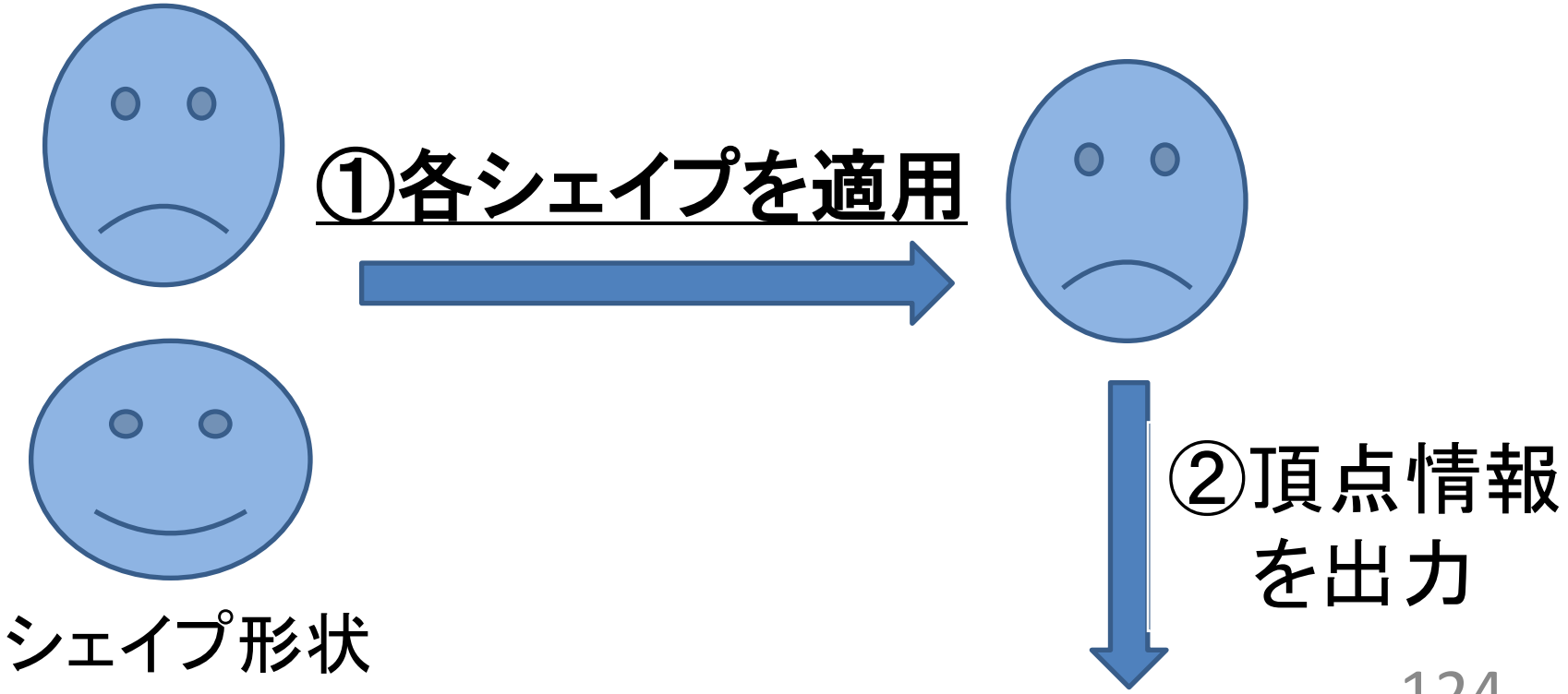
- CGeometryAccessor::**GetShapeKeys()**



～シェイプ情報の取得～

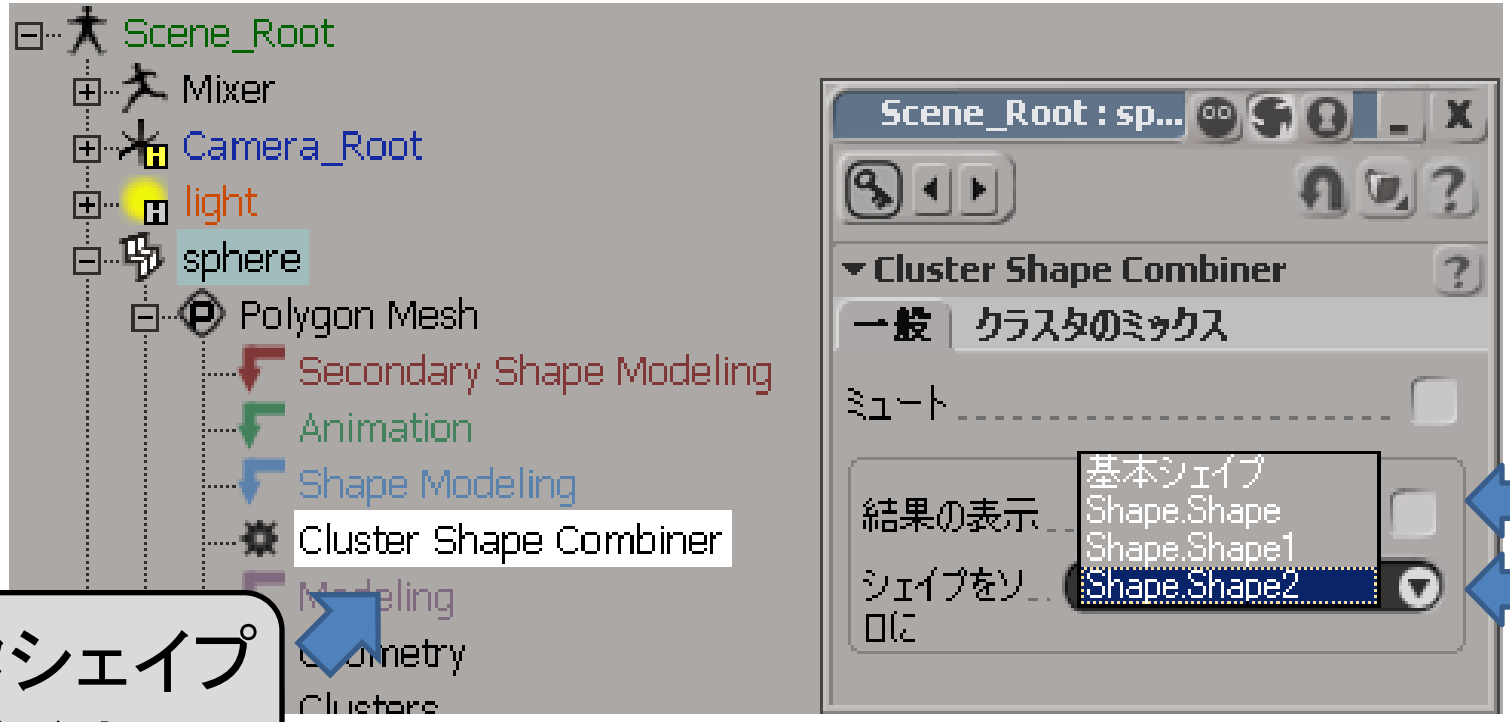
シェイプの頂点情報の取得

1. 各シェイプをポリゴンメッシュに適用
2. 頂点情報を出力



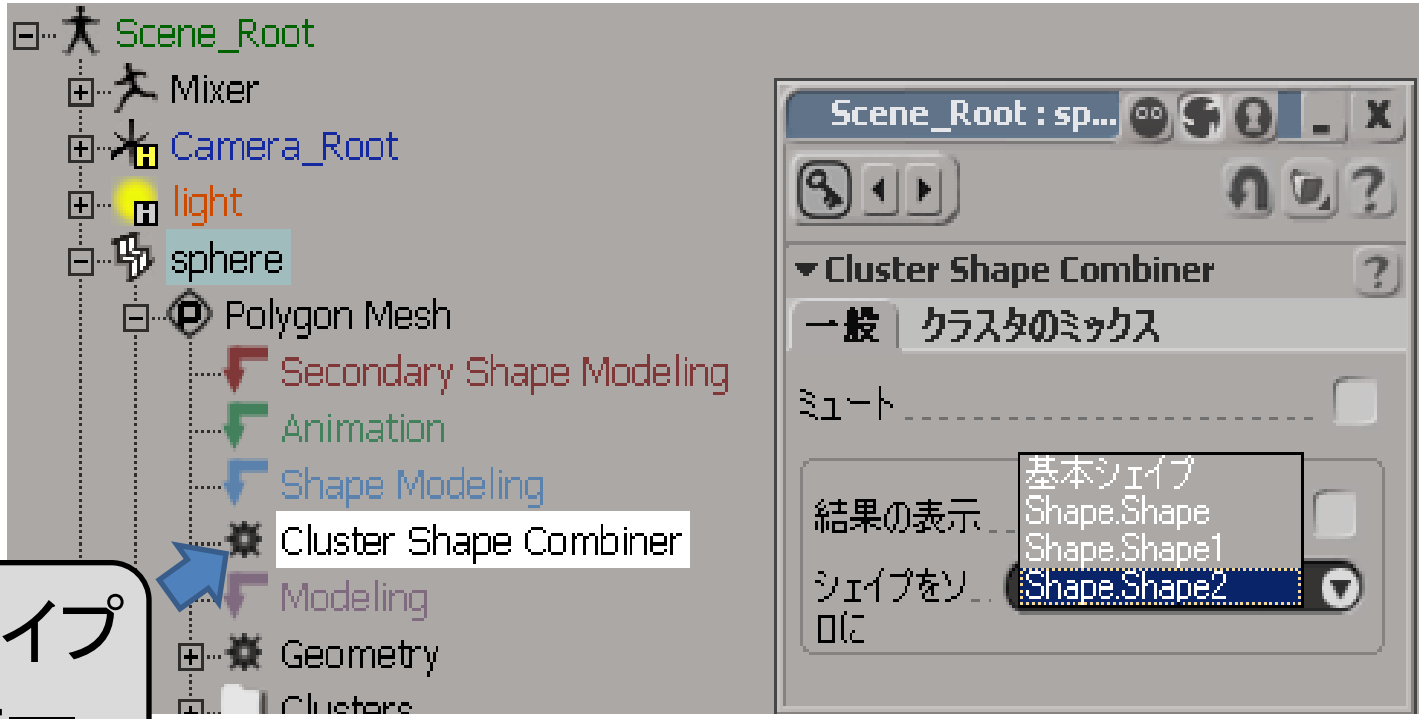
～シェイプ情報の取得～
シェイプの適用（GUI上）

- Softimage上でシェイプを切り替え
- **“クラスタシェイプコンバイナー”オペレータ**



～シェイプ情報の取得～
シェイプの適用

- 今度はプログラムから



クラスタシェイプ
コンバイナー

～シェイプ情報の取得～
シェイプの適用

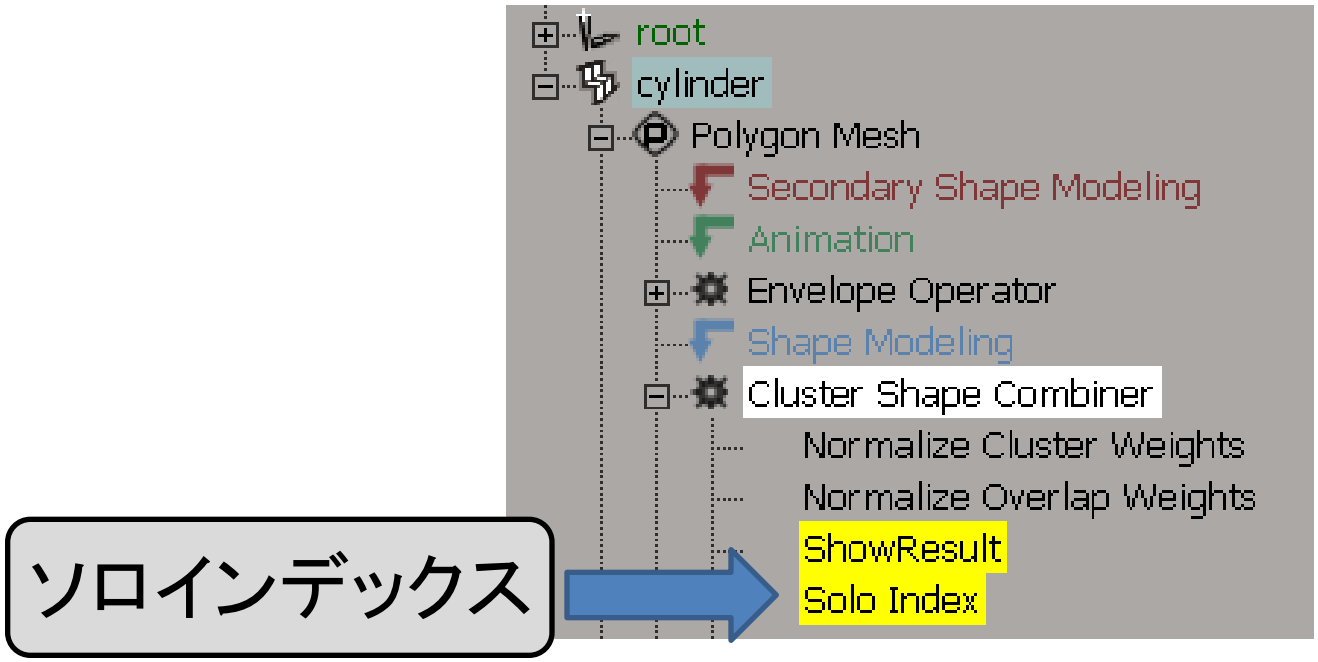
- ”クラスタシェイプコンバイナ”

→ **Operatorクラス**

```
CRef ref;
CString n = x3DObject.GetFullName();
ref.Set( n + L".polymsh.clustershapecombiner" );
Operator sc( ref );
```

～シェイプ情報の取得～
シェイプの適用

- **”SoloIndex”パラメータ**で切り替える



～シェイプ情報の取得～
シェイプの適用

- 同様の操作をプログラムで行う

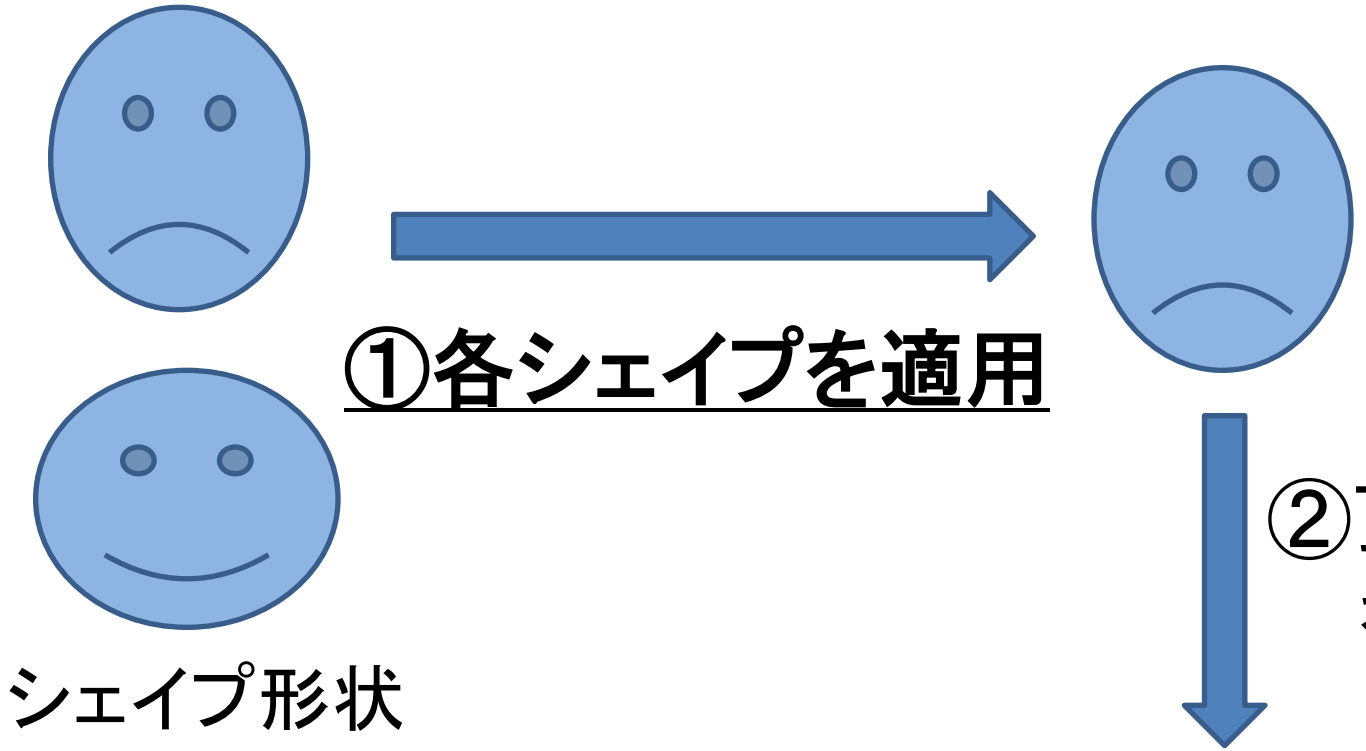
```
Parameter soloIndex;
soloIndex = sc.GetParameter( L"SoloIndex" );

For( int i=0; i<shapeCount; i++ ) {
    // 形状切り替え
    soloIndex.PutValue( i );
    /* 頂点情報を出力 */
}
```

～シェイプ情報の取得～
シェイプの適用

- 各シェイプを適用

→ 各シェイプの頂点情報を出力できた



～シェイプ情報の取得～

まとめ

- シェイプとは？
- 出力する情報
- シェイプマネージャー
- シェイプの取得方法
  - 名前
  - 形状
  - スキニング＋シェイプの場合

カスタムパラメータ情報の取得

- ゲーム用のパラメータを設定する方法
- カスタムパラメータとは？
- データ取得

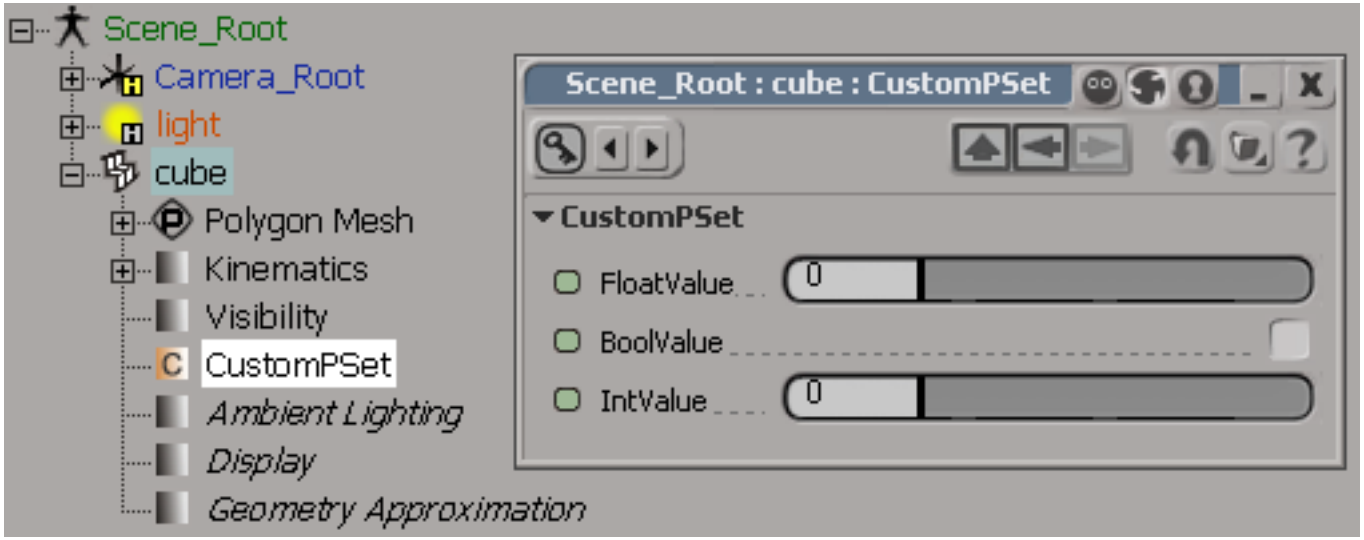
～カスタムパラメータ情報の取得～
ゲーム用のパラメータを設定する方法

- 「**ゲーム用**」のパラメータを設定
  - 外部のゲームエンジンで設定
  - DCCツール上で設定
    - 名前に埋め込む
    - **カスタムパラメータによる方法**
    - その他

～カスタムパラメータ情報の取得～

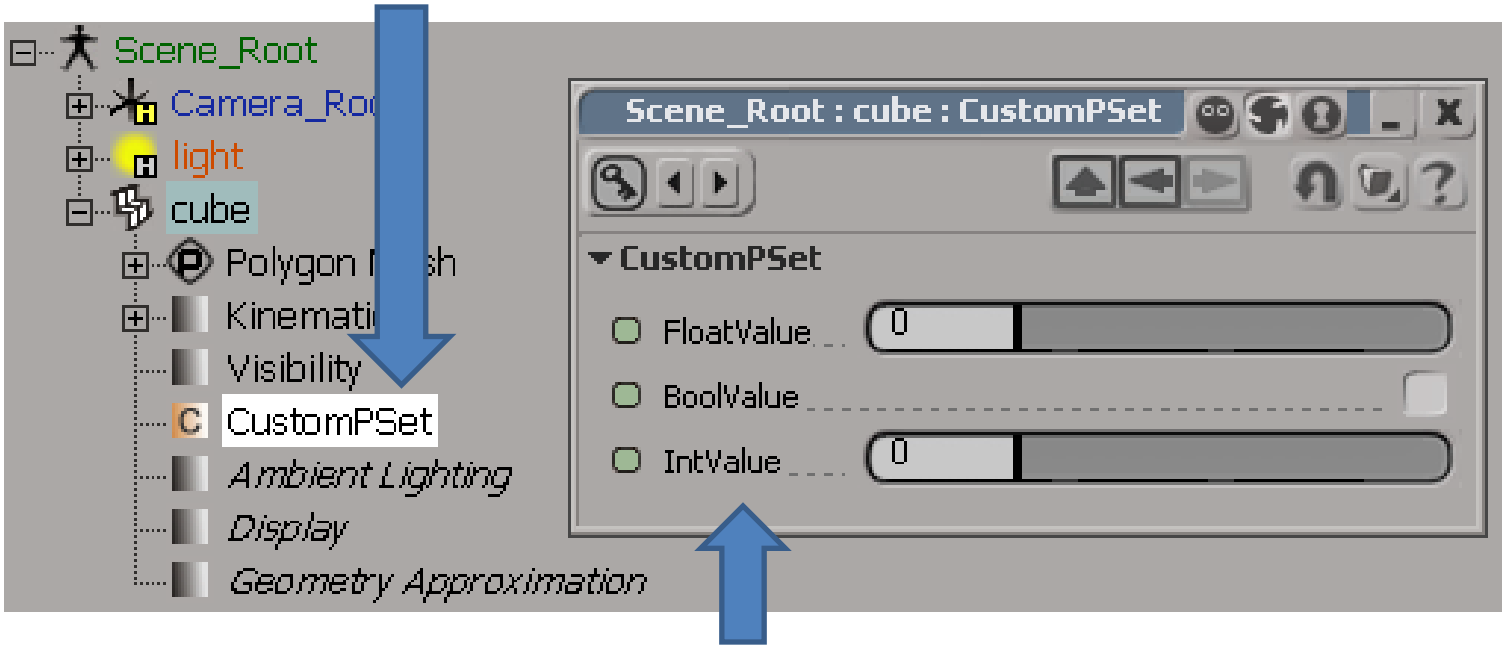
カスタムパラメータとは？

- ユーザーが追加できるパラメータ
- ノード、マテリアル、テクスチャ、イメージソースなどに追加できる



～カスタムパラメータ情報の取得～
カスタムパラメータの構成

カスタムパラメータセット



カスタムパラメータ

～カスタムパラメータ情報の取得～

カスタムパラメータの取得

- プラグインからカスタムパラメータを取得
  1. **カスタムパラメータセット**を取得
  2. **内部のカスタムパラメータ**を取得

～カスタムパラメータ情報の取得～

カスタムパラメータセットの取得

- カスタムパラメータセットは**Propertyの一種**
- SceneItem::**GetProperties**()から取得

```
CRefArray properties = sceneItem.GetProperties();

for( int i=0; i<properties.GetCount(); i++ ) {
    Property prop = properties[ i ];
    if( prop.GetClassID() == siCustomPropertyID ) {
        /* カスタムパラメータセットの情報を取得 */
    }
}
```

～カスタムパラメータ情報の取得～

カスタムパラメータの取得

- 内部のカスタムパラメータを取得
- ProjectItem::**GetParameters**()

```
CParameterRefArray params;
params = property.GetParameters();

for( int p=0; p<params.GetCount(); p++ ) {
    Parameter param = params[ p ];
    /* カスタムパラメータの情報を取得 */
}
```

～カスタムパラメータ情報の取得～

パラメータの型

- カスタムパラメータはユーザーが追加

  → エクスポーターは**パラメータの型を知らない**
- **型情報**は**GetValueType**()で取得
- 戻り値はDataType
  - カラー：siColor4f
  - ３２ビット符号あり整数：siInt4
  - 実数(float)：siFloat
  - ブール型：siBool
  - 文字列型：siString

～カスタムパラメータ情報の取得～

パラメータの値

- 型がわかれば値を取得できる
- **GetValue**()で値を取得

```
DataType type =  param.GetValueType();
if( type == siFloat ) {
    float value = param.GetValue();
}
Else if( type == siString ) {
    Cstring value = param.GetValue();
}
```

～カスタムパラメータ情報の取得～
まとめ

- ゲーム用のパラメータを設定する方法
- カスタムパラメータとは？
- データ取得

エクスポーター作成のコツ

- Softimageの操作を覚えよう
- シーンを小さくしてからデバッグしよう

～エクスポーター作成のコツ～

Softimageの操作を覚えよう

- 基本的な操作を覚えよう
  - テストデータは自分で作る
  - デザイナーに頼まない
- 勉強方法
  - Softimage付属のチュートリアル
  - 社内研修

～エクスポーター作成のコツ～
シーンを小さくしてからデバッグしよう

- デザイナーからもらう**シーンは大きい**

  → デバッグに時間がかかる
- **Softimage上で原因を絞り込む**
  - 問題が起こる**最小限のシーン**を用意する
  - 半分 → 半分 → 半分 → ・・・

まとめ

- SoftimageSDKの基本
- エクスポーターの概要
- ノード
- ジオメトリ
- マテリアル
- アニメーション
- スキニング
- シェイプ
- カスタムパラメータ
- エクスポーター作成のコツ

セッションの資料について

- プレゼン資料
- ドキュメント（PDF）
- エクスポーター本体のソースコード

質問のある方は？

- ご意見、ご質問等はこちらへ

- 株式会社セガ　AM開発技術部
- 立福　寛（たてふく　ひろし）
- Tatefuku_Hiroshi@sega.co.jp
- TATEXH(Twitterアカウント）