

# Webcast 音声のご利用方法

電話でご利用できます

## 電話

オーディオ会議(有料) : 03-3298-4797

オーディオ会議(無料) : 0034-800-900364 (注)

参加コード : 119949

(注) オーディオシステムは、AT&Tを使用しており、無料ダイヤルは、NTT以外のキャリア、IP電話、携帯電話などでは使用できません。

電話会議ログイン後は\*(米印)6を押して電話をミュートにしてください。

# Autodesk® FBX® SDK

## SDKオブジェクトモデル



# はじめる前に

## LiveMeeting と音声会議 の使用方法

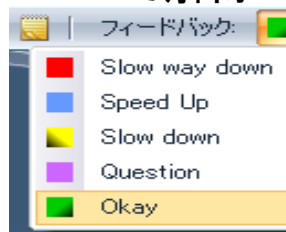
フルスクリーン モード



フルスクリーンに拡張  
ESC で解除



フィードバック



リアルタイム Q&A



電話をミュートする \*6



# 本日のプレゼンテーション

録画がポストされます

ADN Extranet

<http://adn.autodesk.com/>



# エクササイズ – ユニット2

- インポータを作成しシーンノードを再起処理し  
FBXファイルのコンテンツを表示



# FBX SDK Webcast Agenda

- Day / Hour 1 – ようこそFBX SDKへ
- Day / Hour 2 – FBX の基礎 (インポート/エクスポート/シーン)
- **Day / Hour 3 – SDKオブジェクトモデル**
- Day / Hour 4 – ジオメトリ
- Day / Hour 5 – マテリアルとアニメーション



# SDK オブジェクトモデル

- コレクション
- プロパティー
- 接続
- ライトとカメラ
- その他のトピック



# コレクション





# コレクション

- コンテナークラスは[FbxCollection](#)クラスより派生
- 例:  
[FbxAnimLayer](#), [FbxAnimStack](#), [FbxDocument](#), と [FbxScene](#)



# コレクション

- メンバーの追加 - [FbxCollection::AddMember\(\)](#)
- メンバーの削除 - [FbxCollection::RemoveMember\(\)](#)
- メンバー数の取得 - [FbxCollection::GetMemberCount\(\)](#)
- メンバーの取得 - [FbxCollection::GetMember\(\)](#)
- メンバーの検索 - [FbxCollection::FindMember\(\)](#)



# プロパティ



# FbxProperty

- プロパティ用のスタンドアローンクラス
- オブジェクト間の接続を管理するメソッドを提供
- [FbxNode](#)の様なオブジェクトはC++のメンバー変数よりもFBXプロパティ([FbxProperty](#))を使用
  - [FbxObject](#) のデータは型が強化されている
- FbxObjectが作成された際にFbxPropertyの静的なインスタンスが初期化される



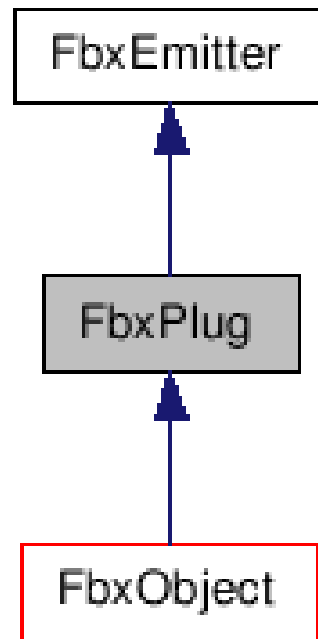
# ユーザー定義データ

- [FbxProperty](#)クラスはユーザー定義データを保持可能
- ランタイムにて[FbxObject](#)に動的に関連付ける



# FbxObject

- オブジェクト間の接続を管理するメソッドを提供
- シーンのオブジェクトの特徴を記述するプロパティー([FbxProperty](#)) メカニズムを提供



# 接続



# 接続

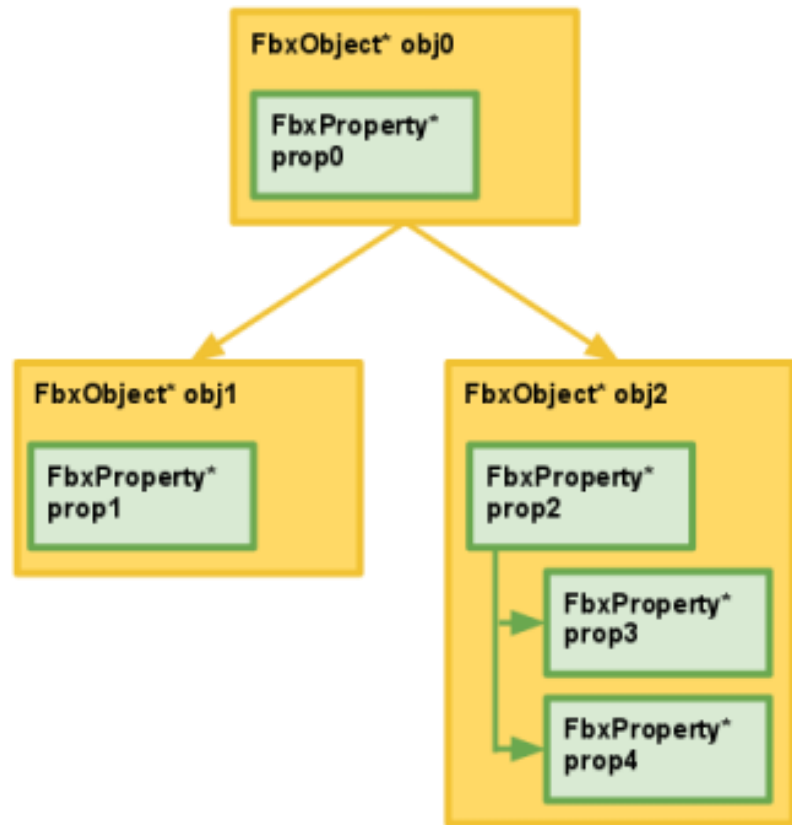
- FBX オブジェクトどうし/又は FBX プロパティ間の2方向の関係を管理する内部データ構造
- [FbxObject](#)と[FbxProperty](#)を使用し操作





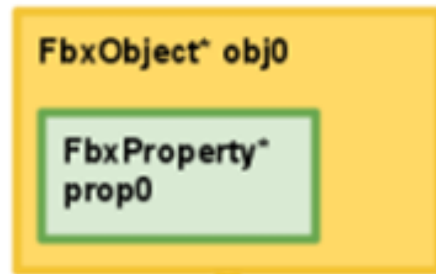
# 接続例

- オブジェクト-プロパティ
- オブジェクト-オブジェクト
- プロパティ-プロパティ



# オブジェクト-プロパティ接続

- オブジェクトがプロパティをソースとして保持
- インデックスより[FbxObject::GetSrcProperty\(\)](#)を使用しオブジェクトのソースプロパティを取得
- 同じく, [FbxProperty::GetDstObject\(\)](#)を使用しプロパティのデスティネーションオブジェクトを取得



# オブジェクト-オブジェクト接続

- オブジェクトの親子関係に接続を使用 (例: シーンのノードハイラルキー)
- 一般的に, オブジェクトの子供がソースで、[FbxObject::GetSrcObject\(\)](#)を使用しアクセス
- オブジェクトの親がデスティネーションで、[FbxObject::GetDstObject\(\)](#)を使用しアクセス



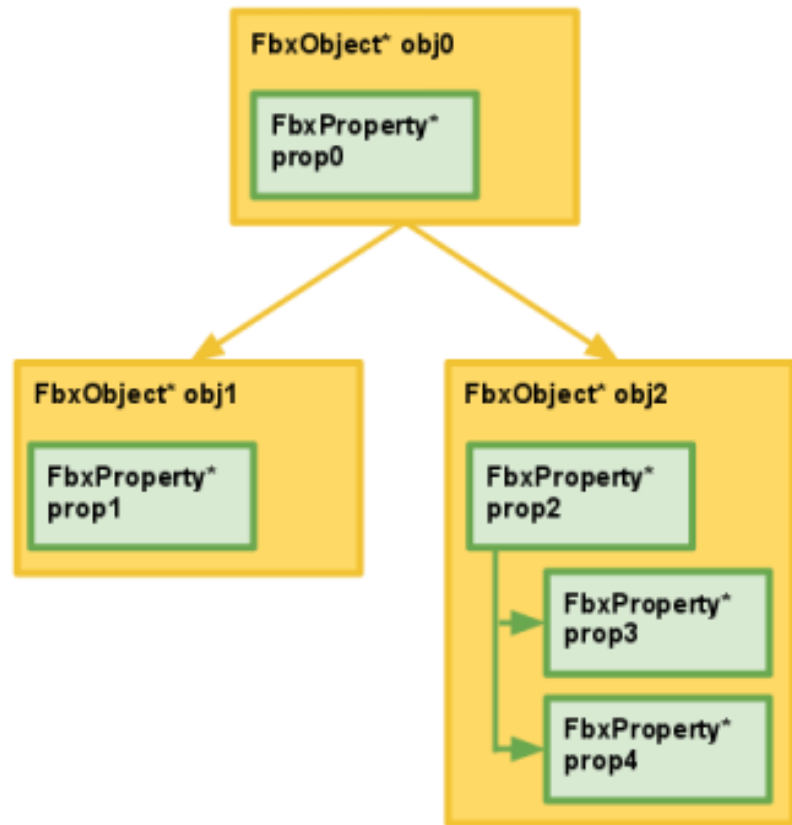
# プロパティ-プロパティ接続

- プロパティの親子関係に接続を使用  
(例: [FbxIOSettings](#)のプロパティハイラルキー)
- 一般的に, プロパティの子供がソースで,  
[FbxProperty::GetSrcProperty\(\)](#)を使用しアクセス
- プロパティの親がディスティネーションで,  
[FbxProperty::GetDstProperty\(\)](#)を使用しアクセス



# 接続例

- オブジェクト-プロパティ
- オブジェクト-オブジェクト
- プロパティ-プロパティ

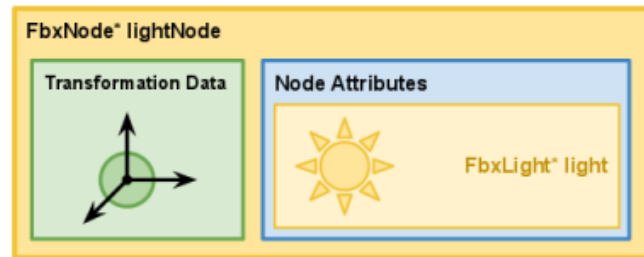


# ライトとカメラ



# FbxLight

- ライト型:
  - ePoint
  - eDirectional
  - eSpot
  - eArea
  - eVolume



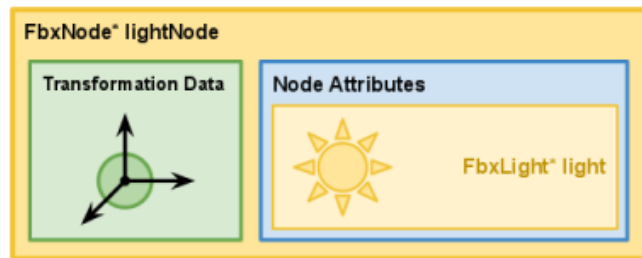
# FbxLight

// スポットライトの作成

```
void CreateLight(FbxScene* pScene, char* pName)
{
    FbxLight* light = FbxLight::Create(pScene,pName);
    light->LightType.Set(FbxLight::eSPOT);
    light->CastLight.Set(true);

    FbxNode* lightNode = FbxNode::Create(pScene,pName);
    lightNode->SetNodeAttribute(light);

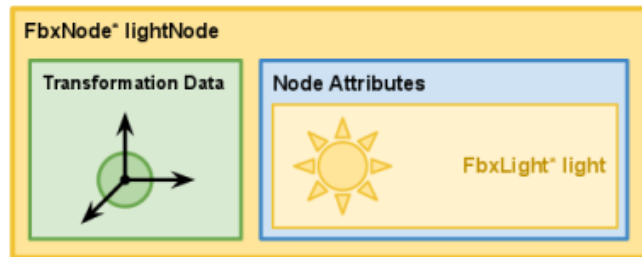
    FbxNode* rootNode = pScene->GetRootNode();
    rootNode->AddChild(lightNode);
}
```



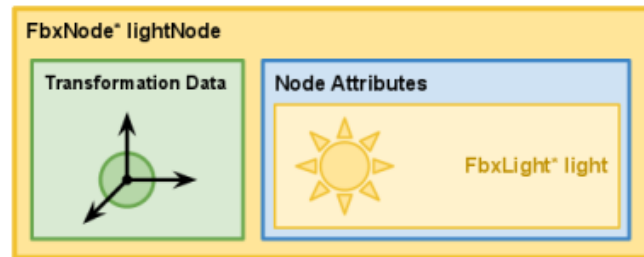


# FbxLight

- Spot/Directionalライトの方向
  - [FbxNode::SetTarget\(\)](#)を使用しターゲットを設定
- Color
  - [FbxLight::Color](#)プロパティーにて定義
- Intensity
  - [FbxLight::Intensity](#)プロパティーにて定義



# FbxLight

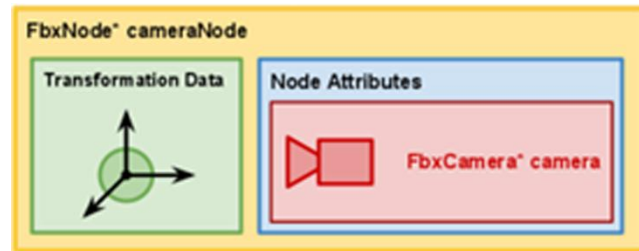


- Decay
  - [FbxLight::DecayType](#) プロパティーにて定義
- Shadows
  - [FbxLight::CastShadows](#) プロパティーを使用し有効に
  - [FbxLight::ShadowColor](#) プロパティーでライトの影の色を定義
  - [FbxLight::SetShadowTexture\(\)](#) を使用し影テクスチャーを設定

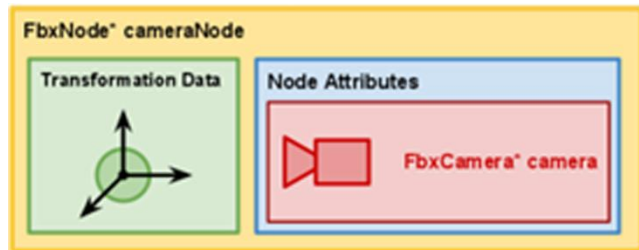


# FbxCamera

- 型:
  - ePerspective
  - eOrthogonal
- FbxCameraStereo
  - ステレオカメラの特性を追加



# FbxCamera



```
void CreateCamera(FbxScene* pScene, char* pCameraName)
{
    FbxCamera* camera = FbxCamera::Create(pScene, pCameraName);

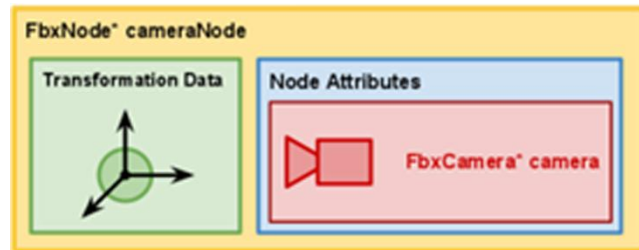
    FbxNode* cameraNode = FbxNode::Create(pScene, pCameraName);
    cameraNode->SetNodeAttribute(camera);

    FbxNode* rootNode = pScene->GetRootNode();
    rootNode->AddChild(cameraNode);

    // カメラが作成されると、シーンの規定カメラとして設定可能
    // シーンにカメラが一つしかなくても、シーンには規定カメラが必要
    pScene->GetGlobalSettings().SetDefaultCamera((char *)camera->GetName());
}
```



# FbxCamera



- カメラの方向
  - [FbxNode::SetTarget\(\)](#)を使用しターゲットを設定
- ドキュメントのFbxCameraを参照しプロパティーを設定
  - 例: Roll, Aspect, FOV, 等



# その他



# FBX オブジェクトの複製

- FBXオブジェクトのメンバー関数Copy()を使用し複製可能
- [FbxProperty](#)のインスタンスや変数が全て複製
- 内部オブジェクトの接続は含まれない



# FBXオブジェクトの複製

// pSceneはシーンオブジェクトのポインター

```
FbxMesh* pSourceMesh = FbxMesh::Create (pScene, "");
```

// ... pSourceMeshのコントロールポイント等を定義

// このメッシュを上書きする

```
FbxMesh* pTargetMesh = FbxMesh::Create (pScene, "");
```

// pSourceMeshからpTargetMeshにデータを複製

// 注意: ソースオブジェクトとターゲットオブジェクトは同じクラスのインスタンスでなければならない (この場合はFbxMesh)

```
pTargetMesh->Copy(pSourceMesh);
```





# エラーハンドリング

- 関数は一般的に失敗した際はfalseを返す
- `objectname->GetLastErrorString()`は文字列でエラーメッセージを返す
- `objectname->GetLastErrorID()`は整数値を返す
  - FBXは、適切にエラーを取り扱うために、enum値又は定義済みの値を使用



# エラーハンドリングのサンプル

- <http://docs.autodesk.com/FBX/2013/ENU/FBX-SDK-Documentation/files/GUID-5509751F-8679-4D32-987D-8343FD8F6D1A.htm>



# サポートされる文字フォーマット

- FBX SDKは内部でUnicode UTF-8を使用
- OSのAPIを使用する際は、UTF-8を必要な文字フォーマットにコンバート
- fbxstring.hのFbxStringが文字列を保持



# SDKのカスタマイズ

- カスタムユーザーデータ
  - FbxObject/FbxPropertyはGet/SetUserDataPtr()を保持
- カスタムプロパティ
- カスタムクラス
  - FbxManager::RegisterFbxClass()
- レイヤー要素用のカスタムユーザーデータ
  - FbxLayerElementUserData
- カスタムファイルフォーマット
  - [Customizing File Formats with FBX SDK I/O Plug-ins](#)



# エクササイズ

- カメラとライトを保持するシーンを作成
- カスタムプロパティーを追加
- シーンの接続を出力
- ASCII FBX ファイルをしてシーンをエクスポート



お疲れ様でした

Autodesk®

# ビデオ

<ftp://sparks:Sparks2012@87.106.99.61/>

又は直接:

IP address: 87.106.99.61

- User name: *sparks*
- Password: *Sparks2012*
- *FBX SDK webcast\_Japanese* フォルダにビデオとサンプルを置きます



# Q&A

[akira.kudo@autodesk.com](mailto:akira.kudo@autodesk.com)