XMLデータベースを使った汎用的データ管理方法について

株式会社ソニー・コンピュータエンタテインメント
JAPANスタジオ　テクノロジー部
大内　慎一

講師紹介

大内　慎一（32）

株式会社ソニー・コンピュータエンタテインメント
JAPANスタジオ　テクノロジー部　R&Dグループ　所属

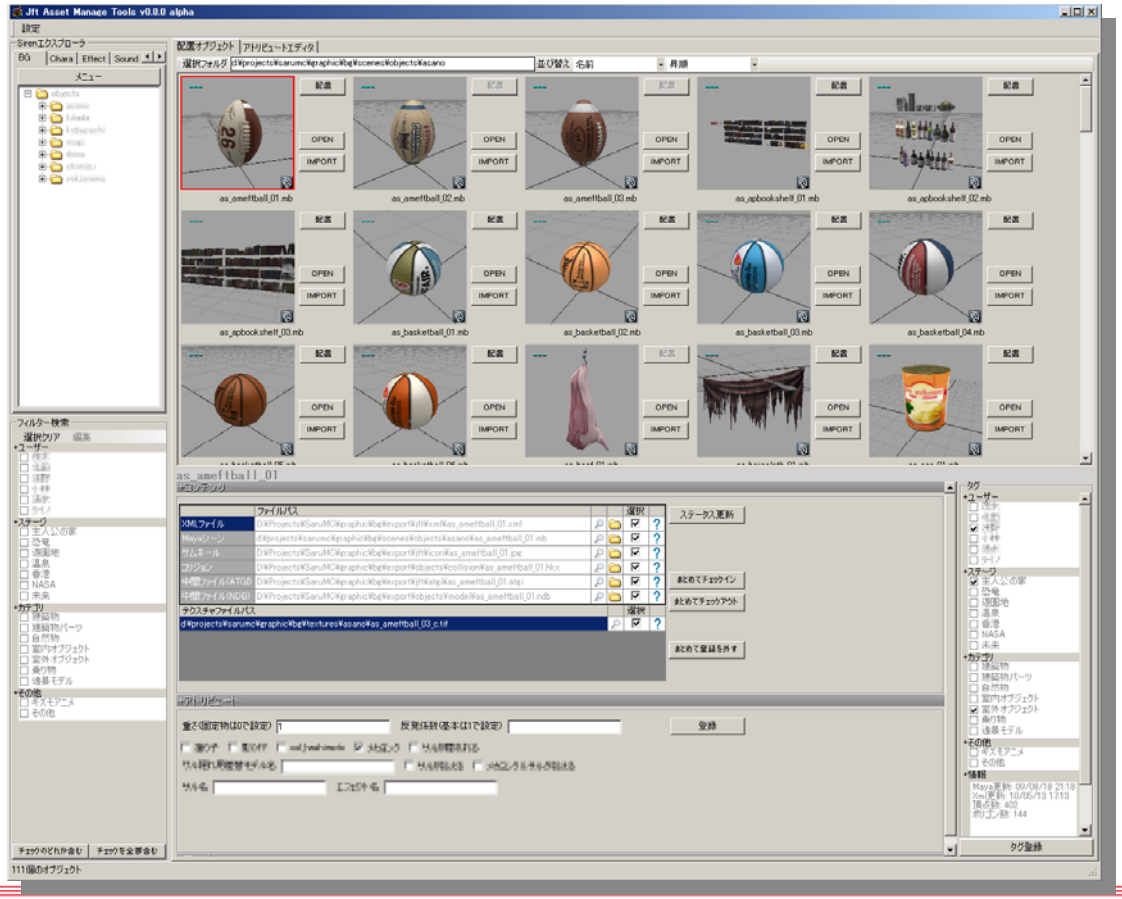
職種：プログラマー（ツール開発担当）
Mayaプラグイン開発（エクスポータ、カスタムマニュピレータなど）
Windowsアプリケーション（アセット管理ツール、分散レンダリングシステムなど）

趣味と性格：家電マニア（電気屋で丸一日過ごせる。ソニー製品好き。そして浪費家）

今日は、**アセット管理ツール**の開発話をもとに
XMLデータベースの紹介をします。

アセット管理ツール

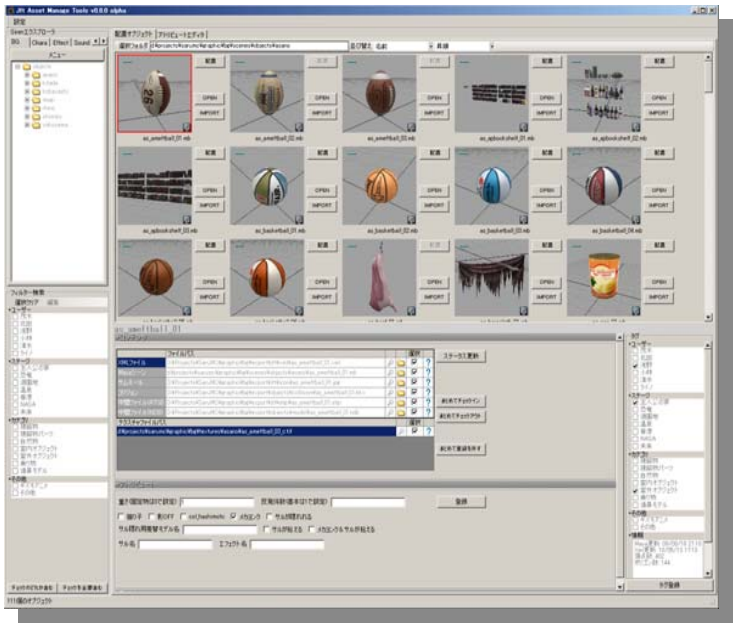
去年、こんな感じのツールを作りました。
動作環境はWindows Xp。開発環境はC# と.NET Framework2.0
使用しているデザイナーさんからJAMTools(ジャムツール)という愛称をもらった。



アセット管理ツール

どんな機能を持っているのか

- オブジェクト配置ツール
- タグ埋め込みとフィルター機能
- アトリビュート設定
- コメント機能
- ファイルサーバへのチェックイン、チェックアウト



これらオブジェクトの管理、制御を

XMLデータベースで行っている

XMLデータベースとは、

XMLファイルを使ったデータベースで、オブジェクトの検索、保存、出力など行う。
SQLは使わない(NoSQL)。XMLのテクニックだけで作れるお手製簡易データベース。

ちなみに、このツールは作成～運用開始いたるまでが**2ヶ月くらい**でした。
弊社にPlacementToolというツールがあり、それを全部作り直す形で作業を進めた。

と言うわけでXMLをネタに話をすすめていきます。

20分と短いので、Q&Aは無し。
会場または交流会で見つけて質問してください。

XML DB vs. SQL DB ― 何故にXMLデータベース

- **XMLはオープンフォーマット**
  - データが壊れてもテキストエディタから直せる。
  - 好きなパース方法が選べる。
  - 出来るだけ完結なフォーマットを策定すれば大掛かりにならずに済む。
  - **コンバータなどを組み合わせたバッチ作業も可能。**
- **現場にSQLが分かるひとが少ない**
  - 作った本人にメンテナンス作業など集中してしまう。引継ぎが難しい。
  - XMLパーサライブラリのほうがもっと簡単に覚えられる。人に薦められる。
- **ユーザーがMySQLなどのモジュールをインストールするのを嫌がる**
  - ライブラリやモジュールが複数にまたがるとメンテナンスが大変。
  - ユーザーはめんどくさがり、セットアップがめんどくさいツールは使ってくれない。

SQLとバージョン管理組み合わせを紹介する開発者は多い。

でも、ゲーム開発のデータ管理は”NoSQL”の手法が向いている

のではないかと

2つのXMLのパース方法について紹介します。

XMLDocument vs. XMLSerializer

.NET Framework を使ったXMLパース方法

例えば、下記のようなXMLフォーマットを定義した場合

```
<?xml version="1.0" encoding="utf-8"?>
<GameObject>
  <Files>
    <File>c:¥C:¥workspace¥cedec¥data¥file01.txt</File>
    <File>c:¥C:¥workspace¥cedec¥data¥file02.txt</File>
    <File>c:¥C:¥workspace¥cedec¥data¥file03.txt</File>
    <File>c:¥C:¥workspace¥cedec¥data¥file04.txt</File>
    <File>c:¥C:¥workspace¥cedec¥data¥file05.txt</File>
  </Files>
</GameObject>
```

XMLDocument

XMLDocumentで書くとこうなる(C#)

```
//XmlDocmentを使ったXmlパーサーコード
class Program
{
    static void Main(string[] args)
    {
        //XMLDocumentクラス作成
        XmlDocument xmlDocument = new XmlDocument();
        //XMLファイルを読み込む
        xmlDocument.Load("C:¥¥workspace¥¥cedec¥¥data¥¥test.xml");
        //ルートドキュメントノードを取得
        XmlElement gameObjectNode = xmlDocument.DocumentElement;
        //ファイルリストノードを取得
        XmlElement fileListNode = gameObjectNode["Files"];
        //ファイルノードのリストを取得
        XmlNodeList fileNodeList = fileListNode.GetElementsByTagName("File");
        //ファイルノードの内部テキストを表示
        foreach (XmlElement fileNode in fileNodeList)
        {
            string value = fileNode.InnerText;
            Console.WriteLine("File名:{0}", value);
        }
    }
}
```

XMLSerializer

XmlSerializerで書くとこうなる(C#)

```
//GameObjectノードに対応したクラス
[XmlRoot("GameObject")]
public class GameObject
{
    //FilesとFileノードに対応したリスト
    [XmlArray("Files")]
    [XmlArrayItem("File")]
    public List<String> Files = new List<string>();
}
//XMLシリアライズのテストコード
class Program
{
    static void Main(string[] args)
    {
        //GameObjectの型を元にXmlSerializerクラスを作成
        XmlSerializer serializer = new XmlSerializer(typeof(GameObject));
        //XMLファイル読み込みのファイルストリームの作成
        FileStream fs = new FileStream("c:¥¥workspace¥¥cedec¥¥data¥¥test.xml", FileMode.Open, FileAccess.Read);
        //XMLファイルからGameObjectクラスのインスタンスを作成
        GameObject gameObject = (GameObject)serializer.Deserialize(fs);
        //ファイルリストの出力
        foreach (String value in gameObject.Files)
        {
            Console.WriteLine("ファイル名:{0}", value);
        }
    }
}
```

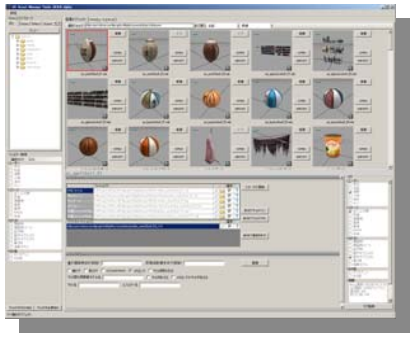
XmlDocument vs. XmlSerializer

- **XmlDocument**
  - すぐに覚えられる。
  - XPath を使った検索ができる。
  - XMLファイルの入出力機能を実装する場合、
    入力と出力の両方のコードを書かなければならない。
- **XmlSerializer**
  - XMLフォーマットをクラスとして考えることができる。
  - XMLフォーマットを出来るだけシンプルに設計したくなる。
  - XmlDocumentと違って、入出力の実装は特に考えなくて済む。

XmlSerializerはオススメ。 ー まわりの人にXmlSerializerの方法を教えたところ、こっちの方が**手間が少なくて**いいね。と言ってくれた。

バージョン管理ツールとの連携

バージョン管理ツールとの連携



アセット管理ツール

チェックイン・登録

チェックアウト・更新

ファイルサーバー

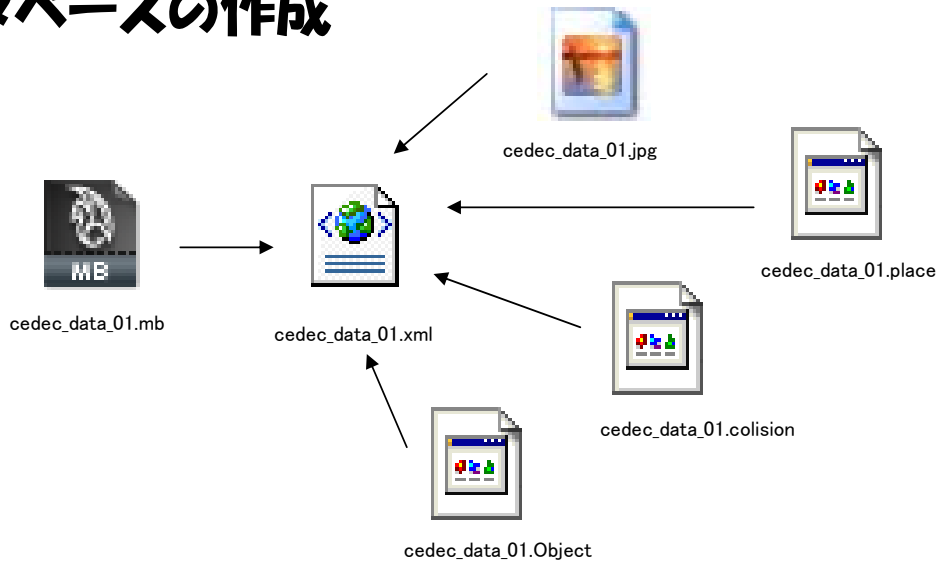
コマンドラインのバージョン管理ツールを使ってファイルサーバーに
チェックイン、チェックアウトなどの作業を行っている。

XMLデータベースを使うことで、関連するファイルをまとめて
チェックイン、チェックアウトすることができる。
※ただし検索といった機能は、ローカルに更新したファイルからしか適用できない。

仕掛けを、大雑把に説明します・・・

バージョン管理ツールとの連携

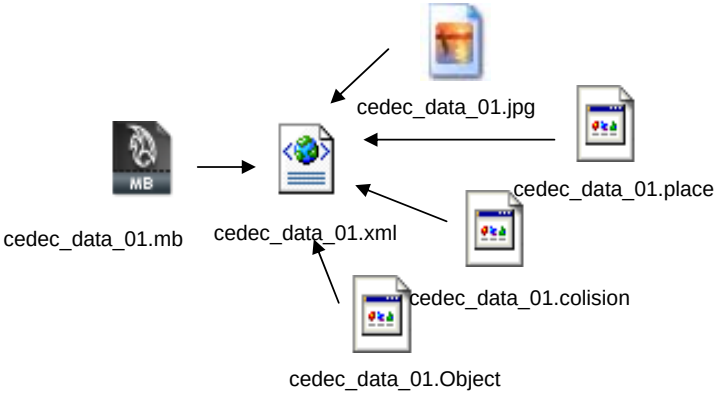
XMLデータベースの作成



1. Mayaバイナリファイル”cedec_data_01.mb”をアセット管理ツールに登録
2. アセット管理ツールは”cedec_data_01.xml”を作成
3. 「cedec_data_01」というキーワードで関連するファイルをフォルダから検索
4. ”cedec_data_01.xml”にそのファイルを登録し保存

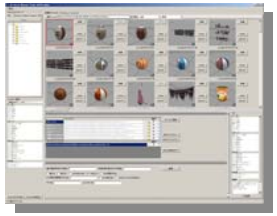
バージョン管理ツールとの連携

アセット管理ツール内部で
XMLファイルはこのように作られる。




```
<?xml version="1.0" encoding="utf-8"?>
<PlacementInfo>
  <GameObject dataVersion="1.1">
    <Name>cedec_data_01</Name>
    <Summary>CEDECでプレゼンするための情報。MBとJPG以外は適当なデータです。</Summary>
    <Keywords>
      <Keyword>大内</Keyword>
    </Keywords>
    <Files>
      <File contentName="Mayaファイル" extension="mb">c:¥workspace¥cedec¥data¥usrname¥cedec_data_01.mb</File>
      <File contentName="サムネール" extension="jpg">c:¥workspace¥cedec¥data¥icon¥cedec_data_01.jpg</File>
      <File contentName="コリジョン" extension="collision">c:¥workspace¥cedec¥data¥collision¥cedec_data_01.collision</File>
      <File contentName="配置データ" extension="place">c:¥workspace¥cedec¥data¥place¥cedec_data_01.place</File>
      <File contentName="実機データ" extension="object">c:¥workspace¥cedec¥data¥model¥cedec_data_01.object</File>
    </Files>
  </GameObject>
</PlacementInfo>
```

バージョン管理ツールとの連携



アセット管理ツール

ファイルサーバー

チェックインや登録の動作

cedec_data_01.mb

チェックイン

cedec_data_01.jpg

チェックイン

cedec_data_01.place

チェックイン

cedec_data_01.colision

チェックイン

cedec_data_01.Object

チェックイン

cedec_data_01.xml

チェックイン

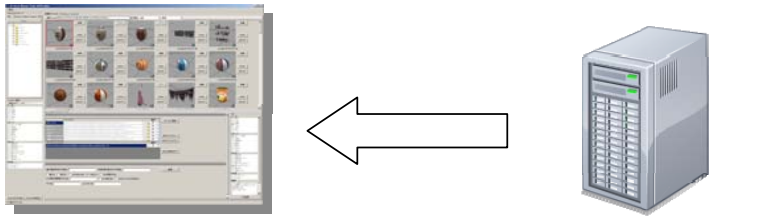
"cedec_data_01"という項目の"チェックインボタン"押す

- cedec_data_01.xmlをパースしFileノードをリストアップ
- 各ファイルにチェックインコマンドを発行
- サーバーに各ファイルが登録、コピーされる
- 最後にcedec_data_01.xmlファイルをチェックイン

cedec_data01.mbに関連するファイルがサーバーにチェックインされる

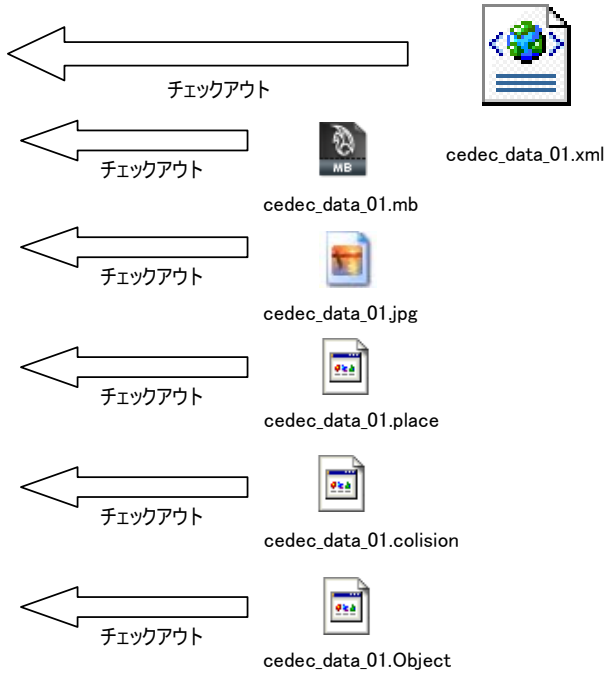
バージョン管理ツールとの連携

チェックアウトや更新の動作



"cedec_data_01"という項目の"チェックアウトボタン"押す

- 最初にcedec_data_01.xmlファイルをチェックアウト
- cedec_data_01.xmlをパースしFileノードをリストアップ
- 各ファイルにチェックアウトコマンドを発行
- ローカルに各ファイルが更新、コピーされる。



cedec_data01.mbに関連するファイルをサーバーからチェックアウトできる

XMLフォーマット定義について


```
<?xml version="1.0" encoding="utf-8"?>
<PlacementInfo>
  <GameObject dataVersion="1.0">
    <Name>cedec_data_01</Name>
    <Summary>CEDECでプレゼンするた
    <Keywords>
      <Keyword>大内</Keyword>
    </Keywords>
    <Maya>c:¥workspace¥cedec¥data¥usrname¥cedec_data_01.mb</Maya>
    <Icon>c:¥workspace¥cedec¥data¥icon¥cedec_data_01.jpg</Icon>
    <Place>c:¥workspace¥cedec¥data¥place¥cedec_data_01.place</Place>
    <Object>c:¥workspace¥cedec¥data¥model¥cedec_data_01.object</Object>
</GameObject>
</PlacementInfo>
```

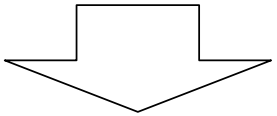
以前は、ファイルフォーマットごとに
XMLノード名を割り当てていた。

例えばここに「**コリジョンデータも管理できるようにしてくれ**」と頼まれたら、
"**Collision**"というノードを増やさなければならなくなる。

このようにXMLノードに特定の意味を与えてしまうと、ファイルファーマットが
増えるたびに、アセット管理ツールのコードに手を加えなければならない。

XMLフォーマット定義について


```
<Maya>c:¥workspace¥cedec¥data¥usrname¥cedec_data_01.mb</Maya>
<Icon>c:¥workspace¥cedec¥data¥icon¥cedec_data_01.jpg</Icon>
<Place>c:¥workspace¥cedec¥data¥place¥cedec_data_01.place</Place>
<Object>c:¥workspace¥cedec¥data¥model¥cedec_data_01.object</Object>
```



```
<Files>
 <File contentName="Mayaファイル" extension="mb">c:¥workspace¥cedec¥data¥usrname¥cedec_data_01.mb</File>
 <File contentName="サムネール" extension="jpg">c:¥workspace¥cedec¥data¥icon¥cedec_data_01.jpg</File>
 <File contentName="コリジョン" extension="collision">c:¥workspace¥cedec¥data¥collision¥cedec_data_01.collision</File>
 <File contentName="配置データ" extension="place">c:¥workspace¥cedec¥data¥place¥cedec_data_01.place</File>
 <File contentName="実機データ" extension="object">c:¥workspace¥cedec¥data¥model¥cedec_data_01.object</File>
</Files>
```

“Files”と”File”という**抽象的**なXMLノードを作って、”**contentName**”と“**extension**”というアトリビュートにファイルの種類を管理するようにした。

このフォーマットの変更によって、ファイルデータの関連情報を**汎用的**に取り扱うことができるようになった。

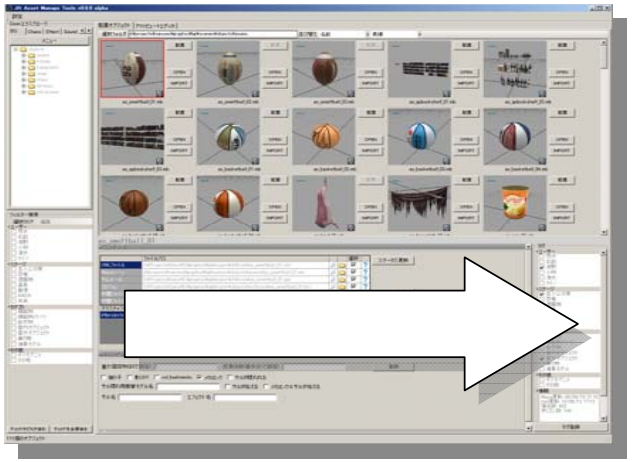
XMLデータベースまとめ

- SQLよりNoSQLの方がシンプル、短時間にデータベースアプリを作れる
  - 紹介したXMLデータベースはNoSQLの一つの手段。他にも方法はある。
  - XMLデータベースはシンプルだから他の人に引き継ぎやすい。
  - 速攻で運用したい場合は、オススメ
- XMLフォーマットは、クラス化を意識して設計する
  - XmlSerializerを使うと、クラスでXMLフォーマットを設計するので
    結果、シンプルになる – **オススメ**
- XMLのノードは**抽象的**に扱ったほうがよい
  - ノードに意味を持たせるとパーサーのメンテナンスが面倒になる
  - アトリビュートを活用して汎用的に扱えば、拡張が楽になる

とりあえずこの講義はココでシメです。
あとは、おまけ！

おまけ！

検索機能



フィルター検索

選択クリア　編集

-ユーザー

+ステージ

- ☐ [redacted]
- ☐ [redacted]
- ☐ [redacted]
- ☐ [redacted]
- ☐ [redacted]
- ☐ [redacted]
- ☐ [redacted]

+カテゴリ

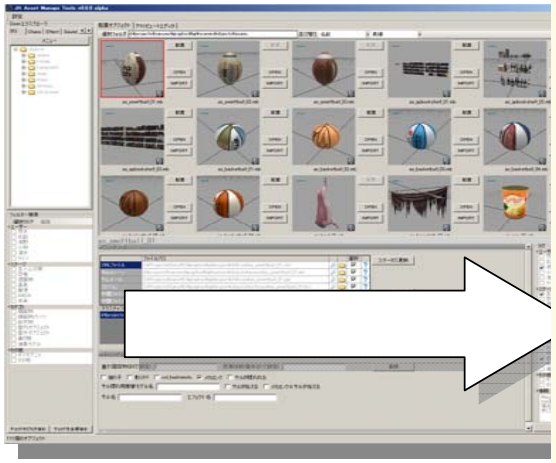
- ☐ 建築物
- ☐ 建築物パーツ
- ☐ 自然物
- ☐ 室内オブジェクト
- ☐ 室外オブジェクト
- ☐ 乗り物
- ☐ 遠景モデル

+その他

- ☐ ギズモアニメ
- ☐ その他

検索機能

データはXmlSerializerでクラスにインスタンス化して検索には、FindAll()を使って行っている



```
//GameObjectノードに対応したクラス
[XmlRoot("GameObject")]
public class GameObject
...
>();
}

// ゲームオブジェクトを検索するクラス
public class FindGameObject
{
    public string Key = "";
    //GameObjectをキーで判定する。
    public bool Match(GameObject obj)
    {
        foreach (string keyword in obj.Keywords)
        {
            if (keyword == Key) return true;
        }
        return false;
    }
}
...
FindGameObject finder = new FindGameObject();
//大内が作成したオブジェクトを検索する。
finder.Key = "大内";
//オブジェクトリストからキーにヒットするオブジェクトのリストを取得する
GameObject[] hitList = Array.FindAll<GameObject>(objects.ToArray(), finder.Match);
```

おまけ！

アトリビュート機能の実装

おまけ！

アトリビュート機能の実装

定義ファイルを読み込んで、
ユーザーインターフェースに反映している

```
<FunctionObject>
  <Name>Attributes</Name>
  <ButtonName>Attributes</ButtonName>
  <Items>
    <Item type="int">重さ(固定物は0で設定)</Item>
    <Item type="int">反発係数(基本は1で設定)</Item>
    <Item type="bool">振り子</Item>
    <Item type="bool">影OFF</Item>
    <Item type="enum" params="人物;建物;エフェクト">種類</Item>
  </Items>
</FunctionObject>
```

おまけ！

自作アセット管理ツール vs. 市販アセット管理ツール

- 市販アセット管理ツールはフルパッケージ
  - 値段が高い。
  - 機能が多すぎる。使い方の分かる人少数になる。
- アセット管理思想が、ゲーム開発と異なる
  - 土木・建築や映像といったジャンルが違うところが発祥
  - ゲーム開発用にカスタマイズが必要
    （自作とカスタマイズのコストは同じになる）
  - バッチ処理が組み込みにくい

もっとゲームに特化したシンプルな管理ツールを開発して欲しい。

タイムアップ？

ご清聴ありがとうございました！

あと、資料作り手伝ってくれたみんなありがとう！