Computer Animation Open Course – Rig and Tools -

# スケールを使ったリグのはなし

## ～DCCツールのしくみからエンジンへの出力まで～



株式会社スクウェア・エニックス
テクノロジー推進部
リードテクニカルアーティスト
**佐々木隆典**

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# この資料について

- セミナーで使用したスライドを pdf 用に改訂しました。
  - 埋め込みで使用した動画は外部リンクに変更。
  - 一部の説明を加筆。
- ページの中の  アイコンは動画を直接開くためのリンクです。
- 以下の資料も合わせてお読み頂くと理解が深まります。
  - CEDEC2014 Technical Artist Bootcamp 2014 vol.2
    ちょっとテクニカルにリギングしてみよう
    https://cedil.cesa.or.jp/cedil_sessions/view/1131
  - 資料閲覧には CEDiL への登録（無料）が必要です。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# 今回お話しすること

- はじめに
- shear 徹底解説
- 様々なツールの scale 仕様 - shear を回避する仕組み
- ゲームランタイムの scale アニメーション - UE4 の例
- Maya® でのリギング
- まとめ

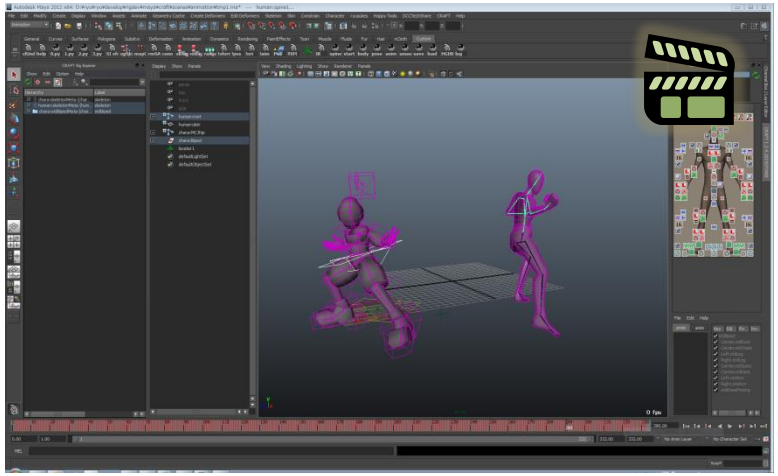
© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

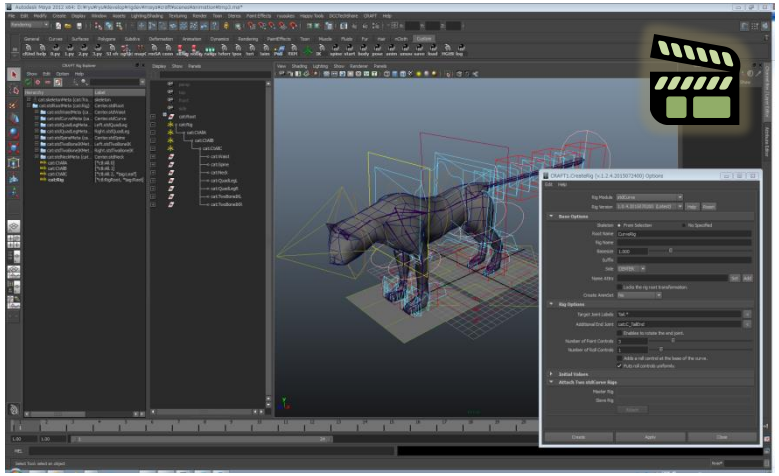
# はじめに



© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# モジュラーリグシステム CRAFT





- 既存のスケルトンに対し、モジュール式で簡単にコントロールリグを作成するツール。
- リターゲットやミラーなど、リグ作成後の様々な補助機能も提供。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# CRAFT の詳細は

より詳細を知りたい場合は、以下の弊社サイトをご参照ください。

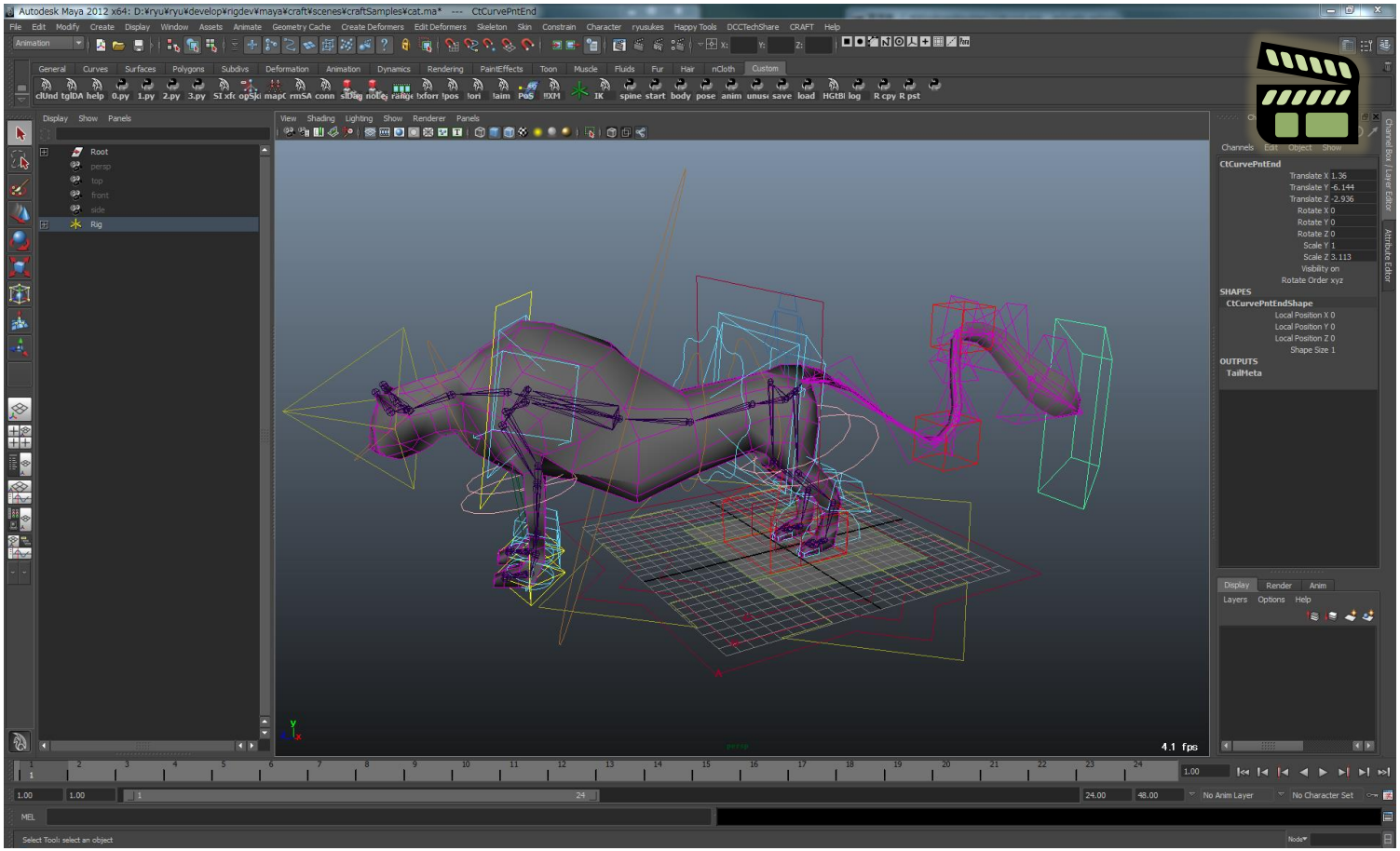
http://www.jp.square-enix.com/tech/publications.html

http://www.jp.square-enix.com/tech/library/pdf/CEDEC2015_CRAFT.pdf

モジュラーリグシステムのアーキテクチャ
Ryusuke Sasaki
CEDEC 2015 (2015)

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# scale リグモジュール




© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# セミナー・アンケートより

- スケールリグの挙動そのもの
- 案件による骨の仕様の違い（セグメントスケール補正の有無）
- スケールしたい軸が joint と違う
- スケールやマトリックス周りの理解
- Softimage®から Maya®に移行したら難しくなった
- 壊れる
- いろいろ不具合が起こる
- 不具合を回避しようとしても難しくなりがち

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# 補足: scale でよく苦労すること

- 互換性の問題
  - ツール間のデータ交換
  - ゲームランタイムとの互換性
- Maya® 上のリギングでも
  - 任意軸での scale が出来ない、又は難しい
  - rotate の取り扱いも難しくなる
  - コンストレインが期待する結果にならない
  - ローカル空間リギングも難しい

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# SHEAR 徹底解説

10

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# scale はなぜ難しいのか

ひとことで言うと…

# shear の存在

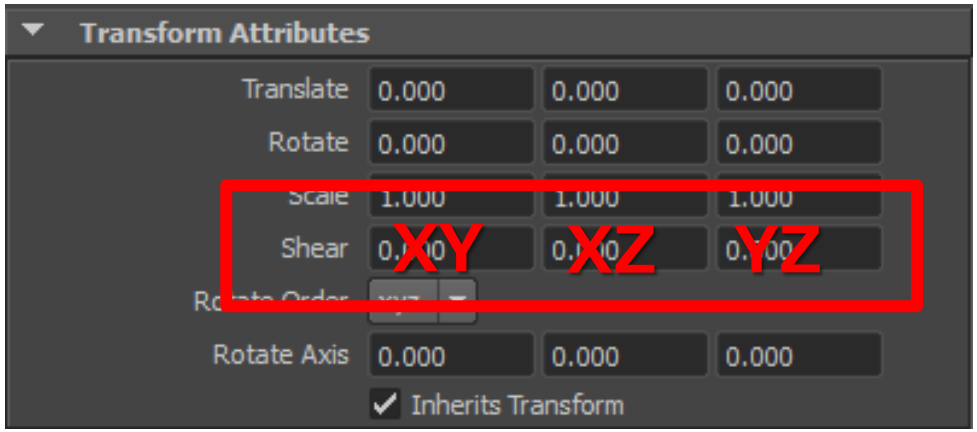
シ　ア　ー

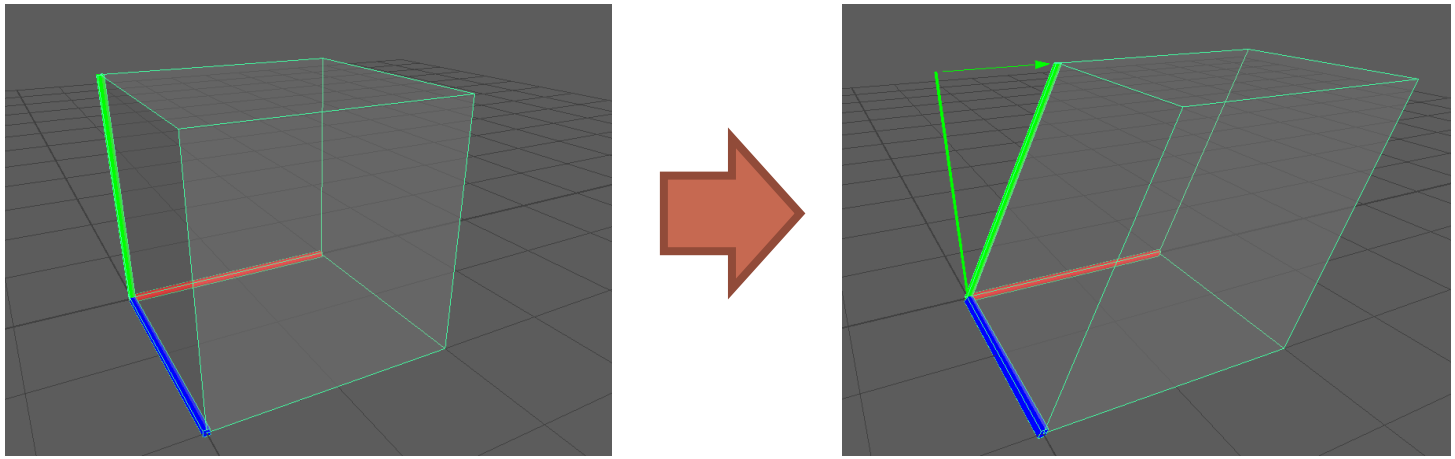
これに尽きる

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# shear ってなに？

Maya® でいうと、これ。

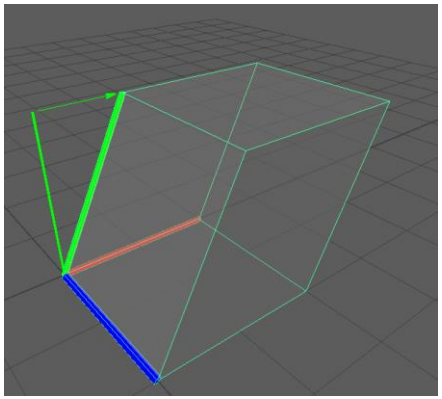


© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

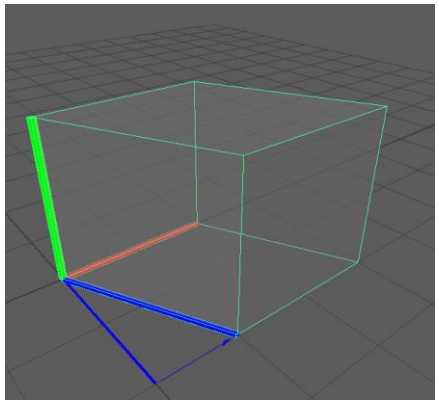
# Maya® における shear の３要素

**xY**



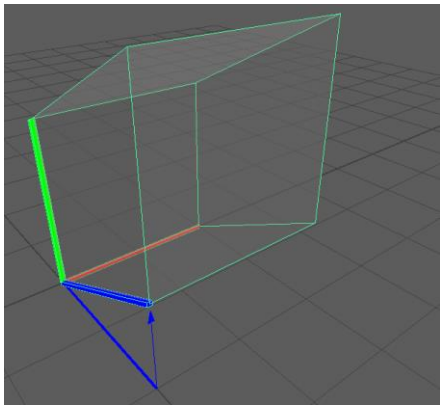
**X軸**に沿って
**Y軸**を傾ける

**xZ**



**X軸**に沿って
**Z軸**を傾ける

**YZ**



**Y軸**に沿って
**Z軸**を傾ける

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

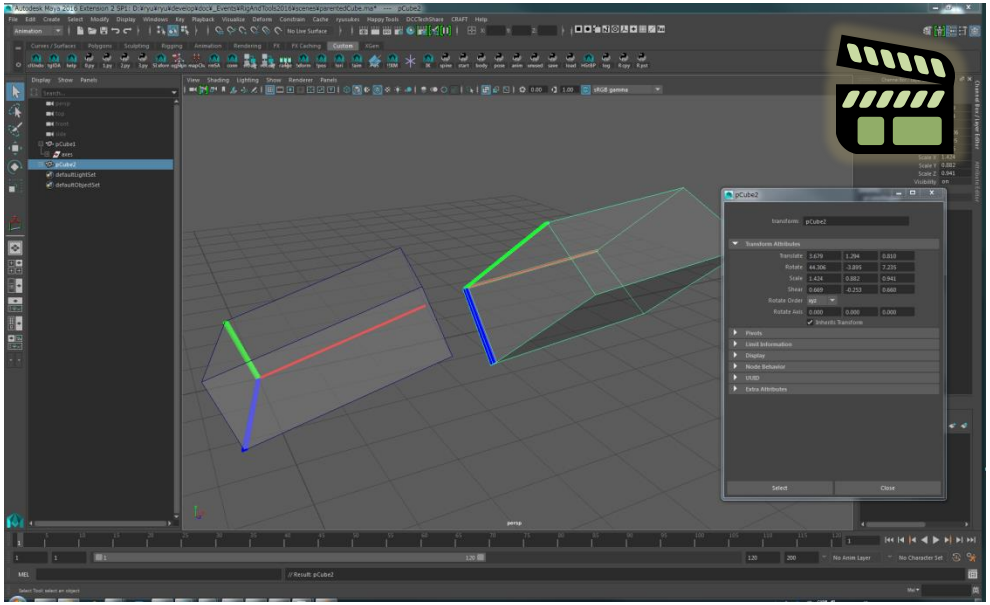
# 軸が足りない？

- X軸を傾けられない？
- Y軸はX軸方向にしか傾けられない？
- 「軸を傾けて形状を歪ませる機能」としてみると、YX, ZX, ZY の３種が足りない。
- 機能として足りない３種を追加することも可能だろうが、shear の重要な役割にフォーカスすると、備わっている３種で必要十分。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# shear の重要な役割

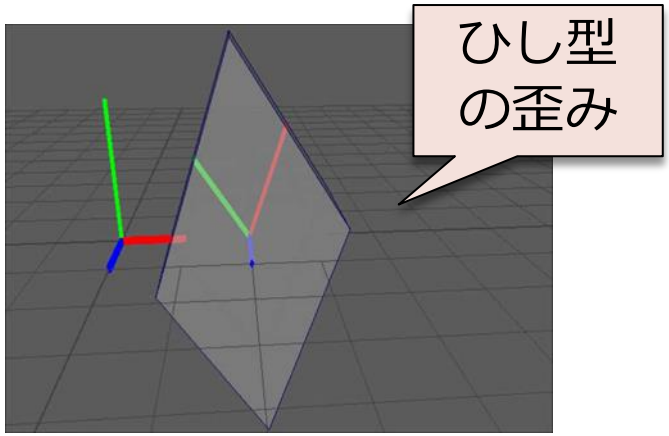
階層構造中のノードのトランスフォーメーション（translate, rotate, scale）を
ワールド空間で評価する場合に必要な要素。



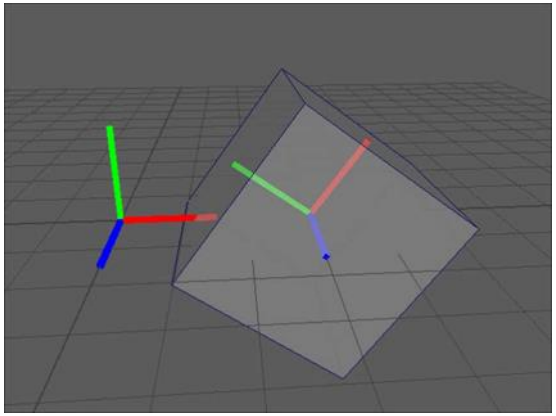
ペアレント解除の例。
shear が無ければ姿勢を維持できない。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

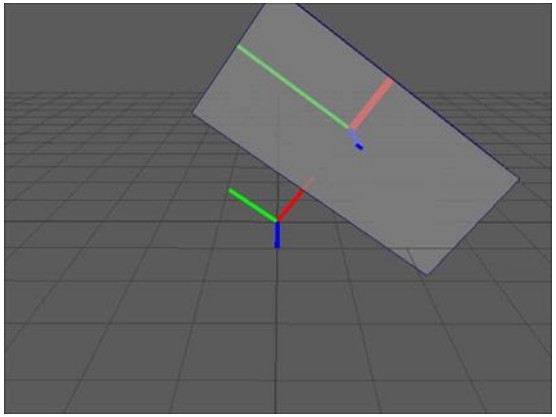
# どういう場合に shear が現れるか



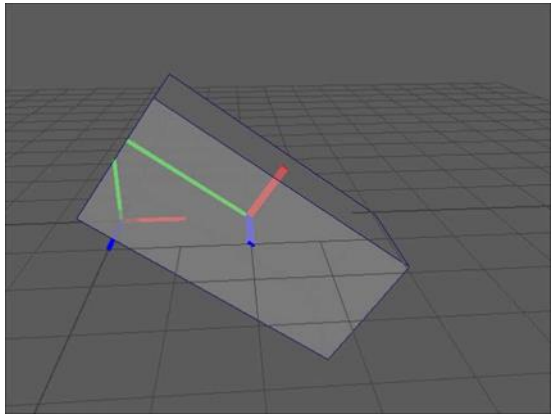

親 不均一scale
子 rotate



親 均一scale
子 rotate



親 rotate
子 不均一scale



子 rotate
子 不均一scale

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# どういう場合に shear が現れるか

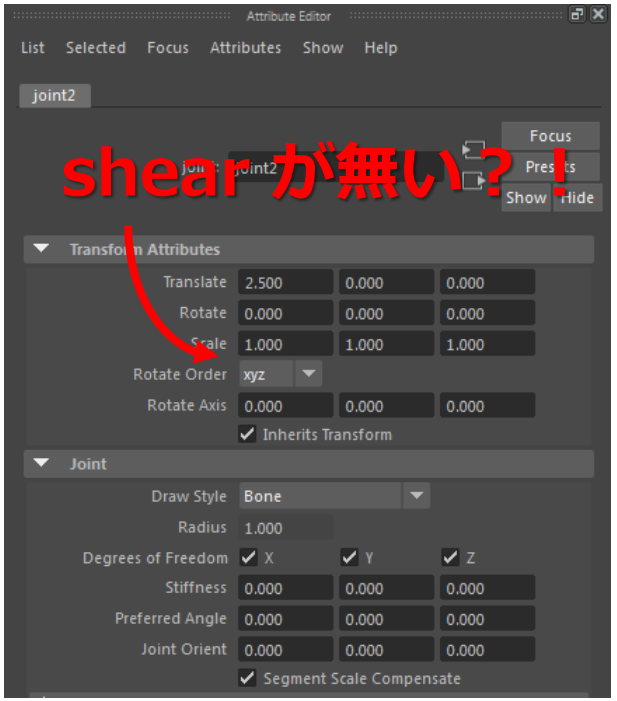
**階層上位で 不均一 scale が使われた場合に、**

**下位の回転で shear が現れる。**

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# joint ノードにおける shear

- joint の Attribute Editor には shear が無い。
- 実際は、shear アトリビュートは存在するので Channel Box に表示させて使うことは可能。
- ちょっとハックだが、そんなに問題はない。

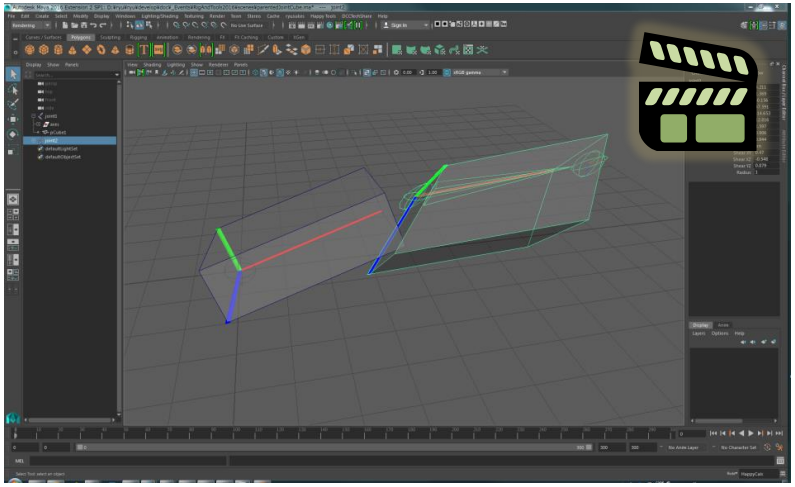



- ちなみに、joint ではピボットポイントも変更することが出来ないのだが、そのアトリビュートも存在はする。しかし実質使用できない。
  一方、 shear は問題なく使える。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# joint ノードの shear の注意点

joint ノードの shear は無いことになっているので、Maya®の様々な機能は shear を回避しようとしたり無視したりする。

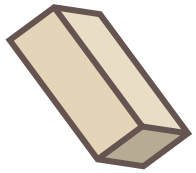


- parent 変更時に shear が必要になる場合、勝手に transform ノードが挟まり shear 無しで姿勢維持される。
- xform コマンドで「shear を含むマトリックス」をセットしようとしても無視される。自分で計算して shear アトリビュートに setAttr すれば大丈夫。

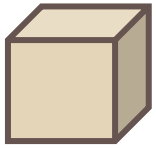
© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# Maya®におけるマトリックス

- ノードのトランスフォーメーションに限定した話…
- ４ｘ４マトリックス
- 乗算は **子 * 親** の順（**左のもの** を **右のもの** で**変換**）



変換されたシェイプ



ローカルシェイプ

**ワールド マトリックス** = **ローカル マトリックス** * **親の ローカル マトリックス** * **親の親の ローカル マトリックス** * ・・・

**ワールド マトリックス** = **ローカル マトリックス** * **親の ワールド マトリックス**

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# Maya® のローカルマトリックスの中身

$$m = sp^{-1} * s * sh * sp * spt * rp^{-1} * ra * r * jo * rp * rpt * is^{-1} * t$$

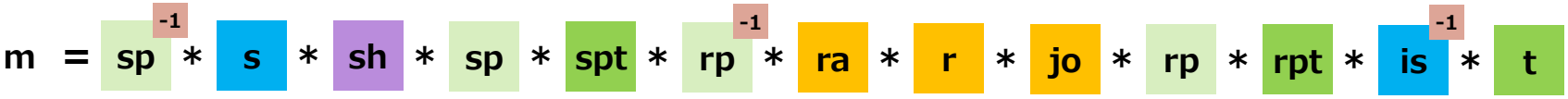
※それぞれはアトリビュートのショート名

joint にしか存在しないもの： jo　is

joint では通常は使われないもの： sh　sp　spt　rp　rpt
（存在はしていて、値を入れれば影響する）

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# Maya® のローカルマトリックスの中身




※それぞれはアトリビュートのショート名

joint にしか存在しないもの： jo is

joint では通常は使われないもの： sh sp spt rp rpt
（存在はしていて、値を入れれば影響する）

translate（色が薄い方はピボット）

rotate

scale

shear

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

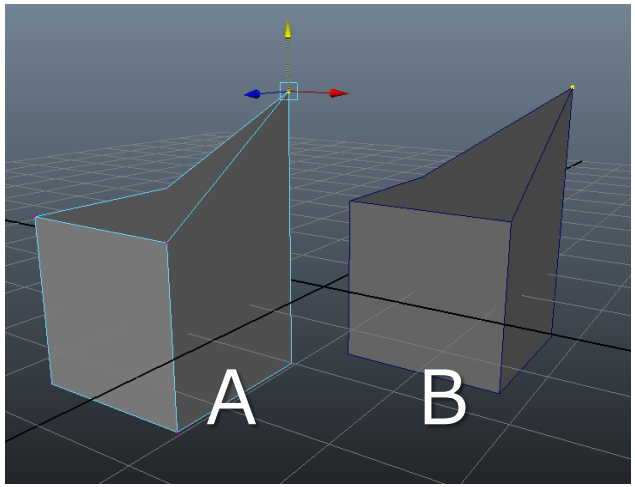
# Maya® のマトリックス出力アトリビュート

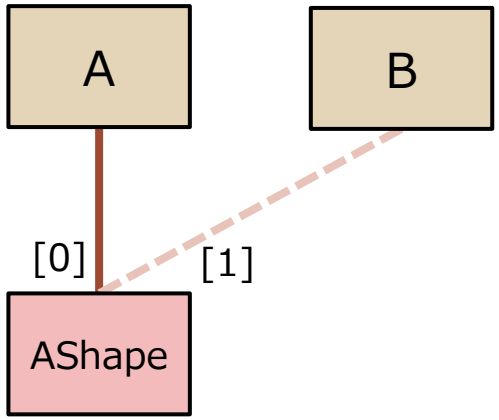
- transformノードは以下のマトリックス出力アトリビュートを持っていて、自由に参照出来る。（括弧はロング名）
  - m （matrix）
  - im （inverseMatrix）

  ローカル系
  - wm[0] （worldMatrix[0]）
  - wim[0] （worldInverseMatrix [0]）

  ワールド系
  - pm[0] （parentMatrix [0]）
  - pim[0] （parentInverseMatrix [0]）

  親のワールド系
- ちなみに、ワールド系に付く [0] はインスタンスのインデックス。



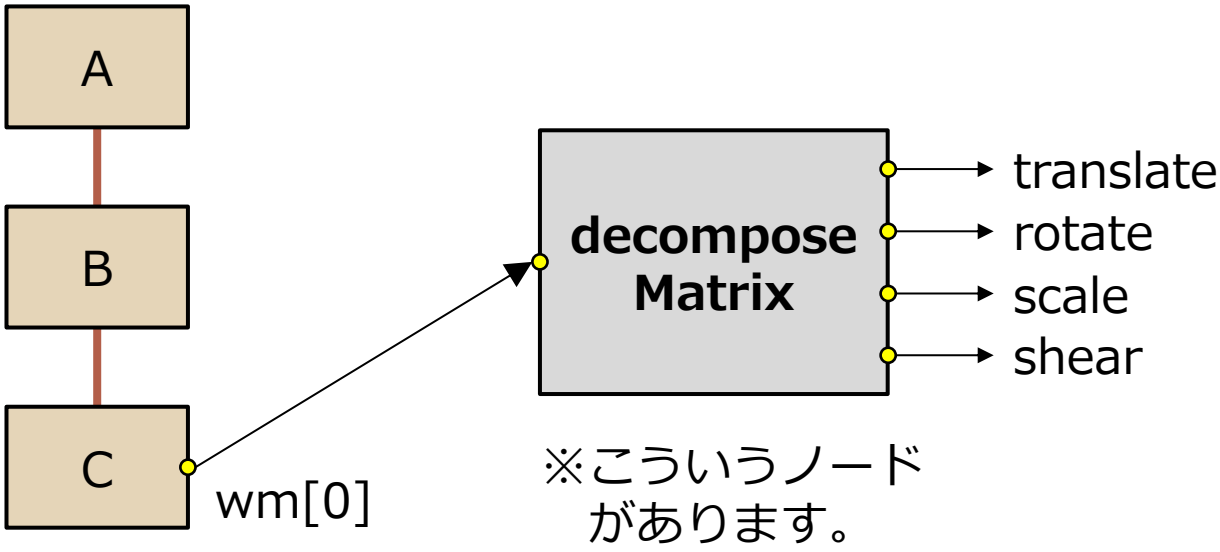

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# マトリックスの分解

- どれだけ乗算されても４要素に分解出来る。

**M = S * Sh * R * T**

Sh (shear) の位置に注意！




- ちなみに、scale=0 を入力しても、極小値で代替される為、分解したり、逆行列を得ることも可能。

© 2014 SQUARE ENIX CO., LTD. All Rights Reserved.

# マトリックス分解のコード

Maya® API を使えば簡単。

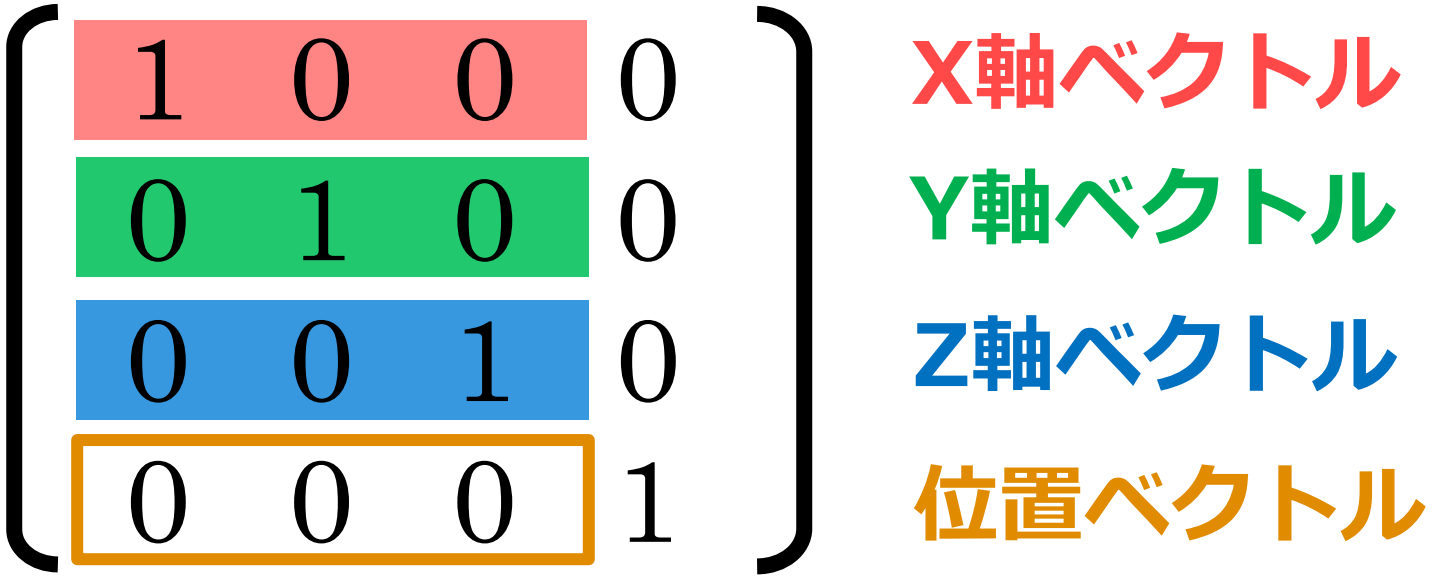
```
import maya.api.OpenMaya as api
sel = api.MGlobal.getActiveSelectionList()
m = sel.getDagPath(0).inclusiveMatrix()
xm = api.MTransformationMatrix(m)
t = xm.translation(api.MSpace.kTransform)
r = xm.rotation(False)  # True なら MQuaternion
sh = xm.shear(api.MSpace.kTransform)
s = xm.scale(api.MSpace.kTransform)
```

自分で実装する場合の参考：

Decomposing a matrix into simple transformations. Spencer Thomas. In "Graphics Gems II", pp 320-323. Morgan Kaufmann, 1991.

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# translate, rotate, scale のマトリックス




行と列が逆の場合もあるが Maya® ではこう。

- shear が無ければ、 3 軸は直交する。
- shear が無ければ、 3 軸それぞれの長さが scale X,Y,Z 。
- 故に scale も無ければ、 3 軸は単位ベクトル。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

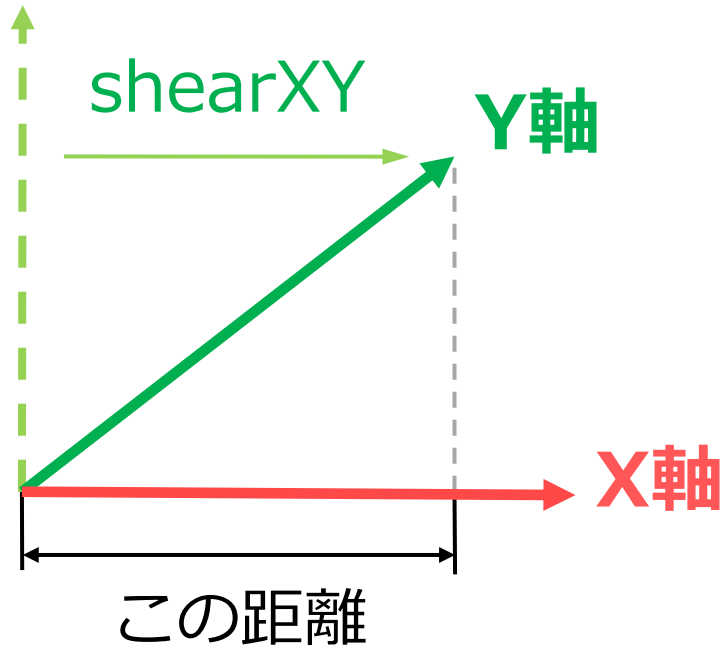
# shear のマトリックス

$$\begin{pmatrix} 1 & yx & zx & 0 \\ xy & 1 & zy & 0 \\ xz & yz & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- 上記のグレーの部分の shear は考えない（ゼロ）。
- rotate, scale, shear は 3x3 部分に混ざるので影響しあう。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# shear 値の意味

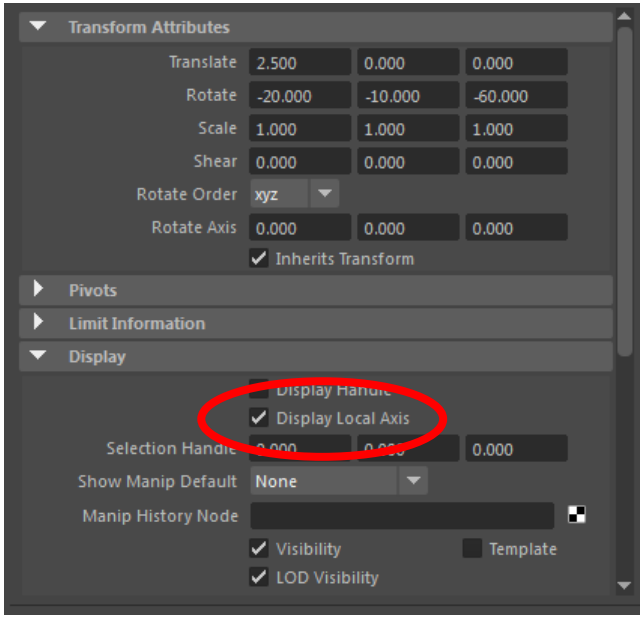



基本的には個々の shear は内積値。
混ざると少々ややこしい（あとでコードを示す）。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# マトリックスの正規直交化

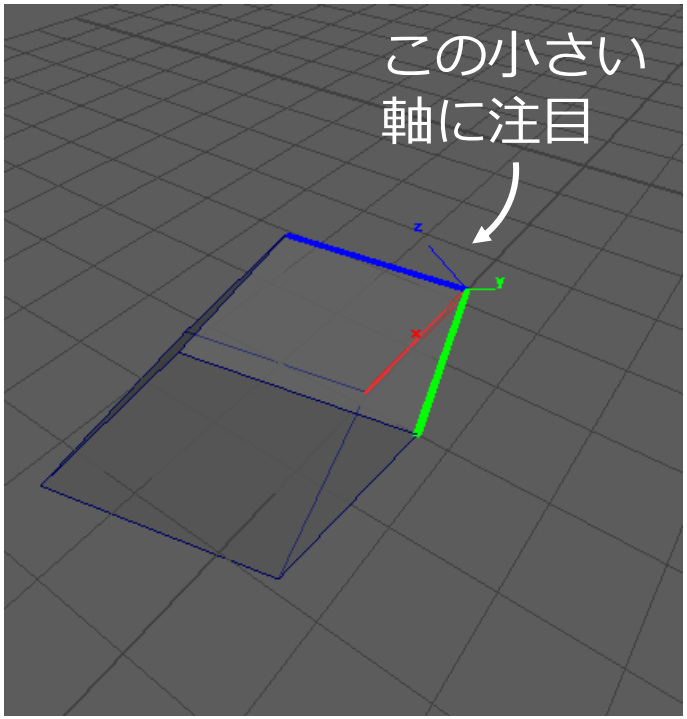
- X軸、Y軸、Z軸の長さが全て 1 で全て直交している状態が rotate (と translate) だけの状態。
- 歪んだ（直交していない） 3 軸がどのように正規直交化されるか = 『マトリックスからどのように scale と shear が除去されて rotate になるか』
- Maya® では、Display Local Axis を on にしてノードの軸を表示すると、 Legacy Viewport で確認出来る（Viewport 2.0 だと歪んだ軸が描画される）。




© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# Maya® による正規直交化を見てみる

- 右図では、形状データとしての歪んだ３軸（大）と、Display Local Axis 機能による直交した３軸（小）を表示している。
- **X軸**は同一方向。
- **Y軸**は、元のXY平面上を維持しつつもX軸に垂直になるように矯正されている。
- **Z軸**は、XY平面に垂直になるように矯正されている。




© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# マトリックス分解の実装（正規直交化の部分）

```
import maya.api.OpenMaya as api
sel = api.MGlobal.getActiveSelectionList()
m = sel.getDagPath(0).inclusiveMatrix()
axisX = api.MVector(m[0], m[1], m[2])
axisY = api.MVector(m[4], m[5], m[6])
axisZ = api.MVector(m[8], m[9], m[10])

sx = axisX.length()
axisX /= sx

shxy = axisX * axisY
axisY -= axisX * shxy
sy = axisY.length()
shxy /= sy
axisY /= sy

shxz = axisX * axisZ
axisZ -= axisX * shxz
shyz = axisY * axisZ
axisZ -= axisY * shyz
sz = axisZ.length()
if axisZ * (axisX ^ axisY) < 0.:
    sz = -sz
shxz /= sz
shyz /= sz
axisZ /= sz
```

マトリックスから
XYZ軸を取得

- scaleX 抽出、X軸を正規化
- shearXY 抽出、Y軸をX軸に直交化
- scaleY 抽出、Y軸を正規化
- shearXZ 抽出、Z軸をX軸に直交化
- shearYZ 抽出、Z軸をY軸に直交化
- scaleZ 抽出、Z軸を正規化

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# 正規直交化と shear パラメータの関係

- Maya® では、X軸 > Y軸 > Z軸 の優先順位で直交化される（私の知る限り多くのツールがそう）。平均などではない！
- shear の値は、この直交化プロセスに基づいているため、XY, XZ, ZY と決まっている。
- もし、Maya® に備わっていない他の shear 値（YX, ZX, ZY）を考えるならば、それに適した軸の優先順位となる。またその場合、直交化の結果が変わるわけなので、分解される rotate 値も変わってくる。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# 補足：負の scale の分解の再現性の問題

負の scale 値を使用したマトリックスを分解しても元の scale 値が得られないことがある。

例えば：

- (-1, -1, -1) を入力、分解結果は (1, 1, -1) となる。
  また、使用ツールによって、その逆パターンもある。
- (-1, 1, 1) や (1, -1, 1) を入力しても、分解結果は (1, 1, -1) や (-1, -1, -1) となる。
- (-1, -1, 1) を入力、分解結果は (1, 1, 1) となる。

分解結果の違いは回転で埋め合わされる。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# 補足：負の scale の分解の再現性の理由

- これは、分解の計算時に「鏡像かどうか」を判別して、軸を反転する処理をするため。つまり二択。
- 鏡像かどうかは『行列式』が負数かどうか。
  負の値を設定した軸の数が奇数個だと鏡像だが、偶数個だと鏡像ではない。
- 鏡像かどうかによって軸が反転され、元の反転軸との違いは回転で表現される。
- 分解時に軸がどう反転されるのかは実装による。
  Maya® では Z 軸のみが反転されるようだ。
  MotionBuilder® では全軸が反転されるようだ。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# 様々なツールの SCALE 仕様

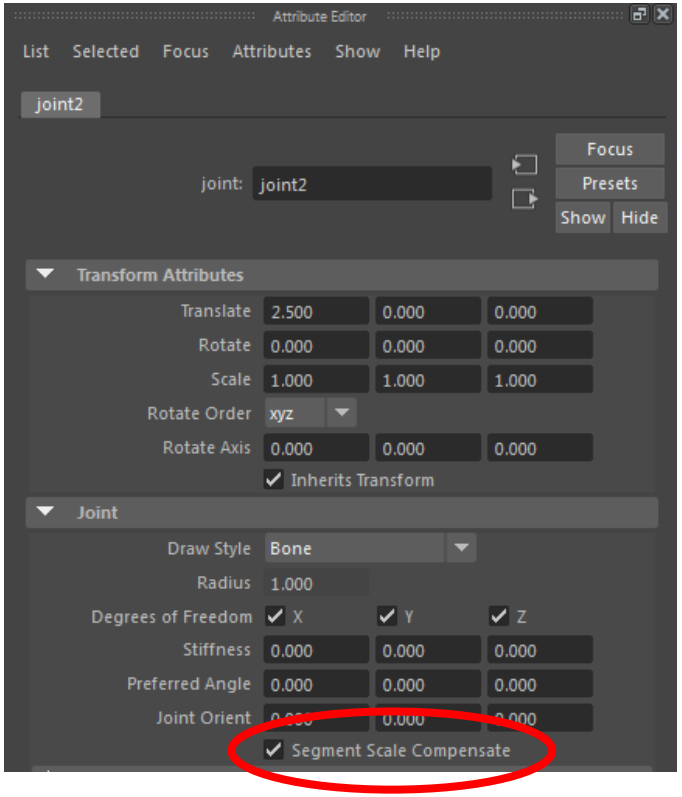
35

## shear を回避する仕組み

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# Maya® の セグメントスケール補正 (ssc)

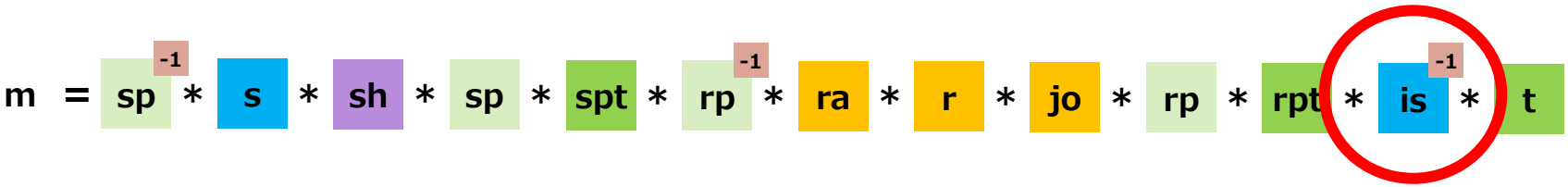
- 親の local scale を打ち消す機能。
- segmentScaleCompensate (ssc) アトリビュートが on の場合に有効。ノードごとの on/off が可能。
- 打ち消されるのは、直接の親の scale のみなので、階層上位のどこかが off だと shear が発生する場合がある。




© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# Maya® の セグメントスケール補正 (ssc)

- 親の scale は inverseScale アトリビュート への入力コネクションとして与えられる。
  ペアレント変更時は自動的にコネクション変更される。
  切断したら機能は働かなくなる。

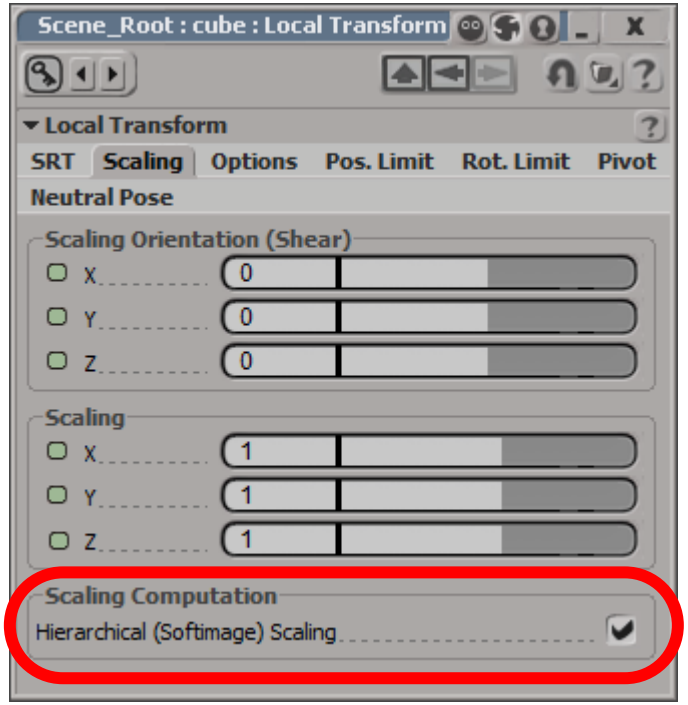



- 親 scale の打ち消しは translate の後なので、親を scale すると子が移動しボーンは伸びる。
  しかし、それ以降には scale は伝搬しない。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# Softiamge® の 階層スケーリング

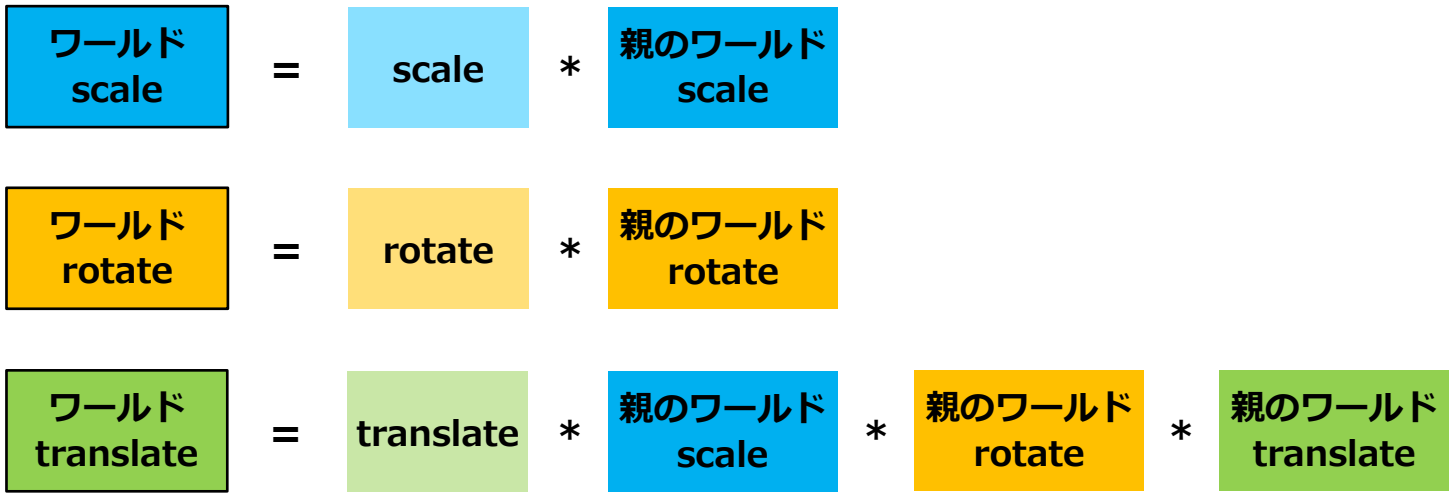
- rotate と scale を混ぜずに別々に乗算する機能。
- Hierarchical Scaling が on の場合に有効。ノードごとの on/off が可能。
- Maya® の ssc と違って子階層に scale が伝搬しつつも、ずっと歪まない。




© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# Softiamge® の 階層スケーリング

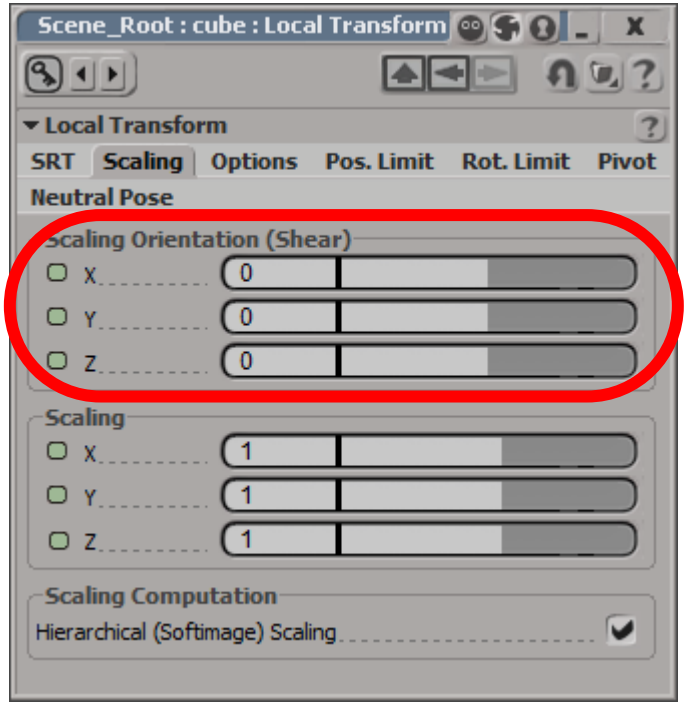
- scale リグのコントロールとして理想的であることが多く、その考え方は Maya® のリギングでも非常に有用。何も低レベルからこの仕様でなくて良い。




© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# Softiamge® の shear と任意軸スケール

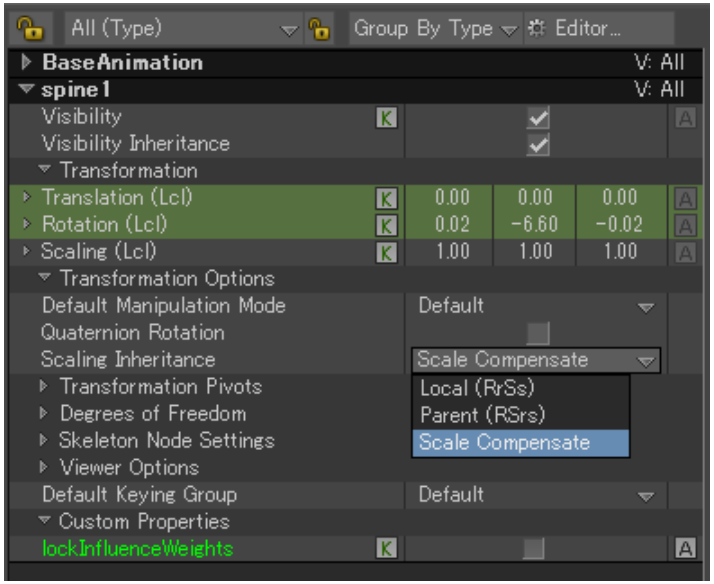
- Scaling Orientation というパラメータが shear の代わりに備わっている（カッコ付きで shear と書かれている）。
- scale の軸のみを回転させて、任意の方向にすることが出来る機能。
- パラメータとしての shear の意味こそ Maya® と違うが、結局は同じことが出来る。




© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# MotionBuilder® の scale 仕様

- shear に相当する機能は無い。よって、ペアレント変更の際に元の姿勢が維持されない場合がある。
- scale の振る舞いは、ノードごとの Scaling Inheritance で設定可能。
  MotionBuilder® は、この設定によって他ツールとの互換性を得ている。




© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# Scaling Inheritance と各ツールの関係

| MotionBuilder® Scaling Inheritance | Local (RrSs) | Parent (RSrs) | Scale Compensate |
|---|---|---|---|
| Maya® Segment Scale Compensate | | OFF | ON |
| Softimage® Hierarchical Scaling | ON | OFF | |
| 3ds MAX® | | ○ | |

3ds MAX® は詳しくないので、もし間違っていたらごめんなさい。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# FBX® の scale 仕様

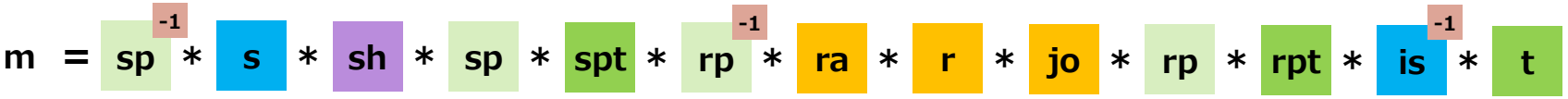
- MotionBuilder® と全く同じ。
- shear は無し。
  Maya® からエクスポートすると shear は出力されずに欠落するので注意！
- “InheritType” という enum プロパティ。
  - 0 … Local (RrSs) ※デフォルト値（省略される）
  - 1 … Parent (RSrs)
  - 2 … Scale Compensate

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# 中間ファイルについて思うこと

当然ながら、ツールによって仕様が異なる。

- scale 仕様の違い
- ピボットの有無、仕様の違い
- その他、様々な補助的な機能の違い




Maya® アトリビュート
有り過ぎ？

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# 中間ファイルについて思うこと

- どれだけ機能が有っても、仕様が違っても、translate, rotate, scale, shear で表現出来る。
- シンプルに考えるなら、
  中間ファイルフォーマットでは shear か matrix 表現をサポートした方が良いと思う。
  ツール間の仕様違いを埋められる。

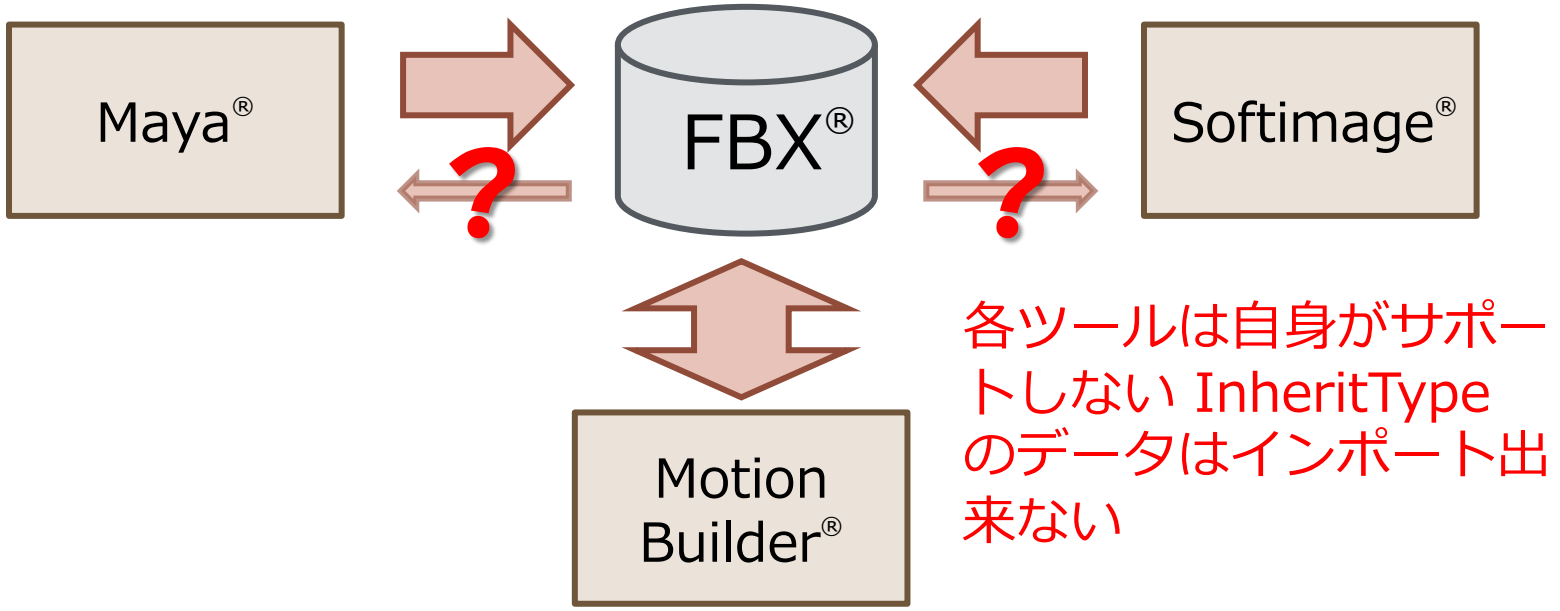
© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# FBX® の由来と特性

- 由来としては、 FBX® は単に MotionBuilder® 専用のファイルフォーマットだったが、データ交換のデファクトスタンダートとなった。
- MotionBuilder® はアニメーション専用ツールとして、他のツールとの互換性を持つ必要があり、InheritType 切り替えによって他のツールの scale 仕様に合わせられるようにした。
- しかし、それは scale 仕様の異なるツール間のデータ交換をサポートするものではない。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# FBX® でのデータ交換




- ツール間での不均一 scale 含むデータ交換は実質不可。
- shear 値は欠落。
- shear 回避という点では InheritType は一理ある。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# ゲームランタイムの SCALE アニメーション

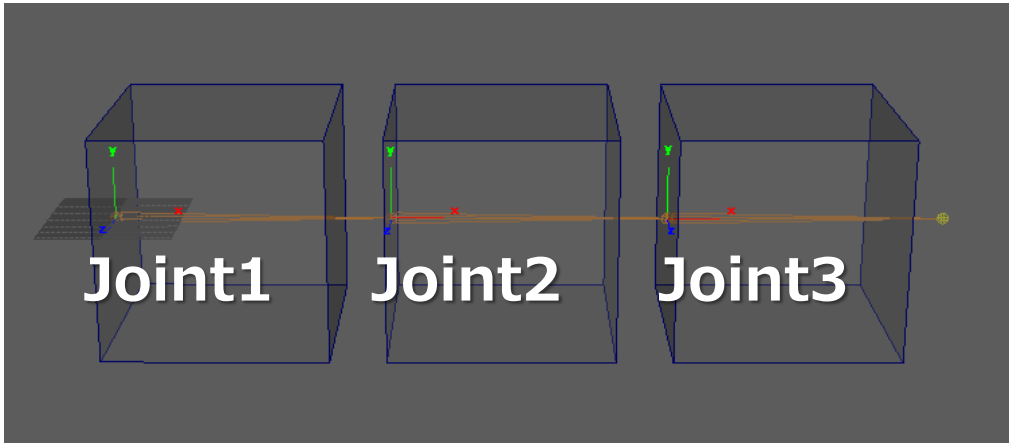
48

## Unreal Engine 4 の例

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# Unreal Engine 4 への FBX® インポート

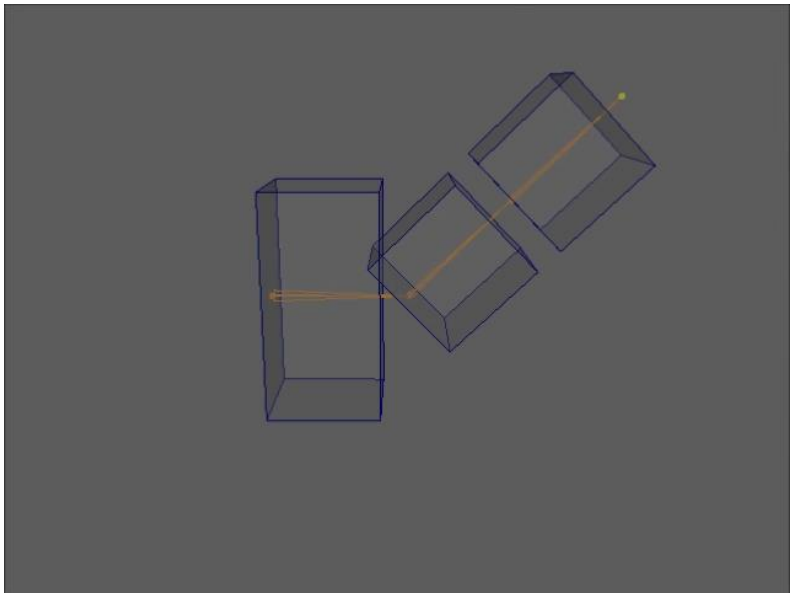
- ３個のキューブがそれぞれの位置のボーンに 100% のウェイトでバインドされた FBX® ファイルを UE4 のスケルタルメッシュとしてインポート。



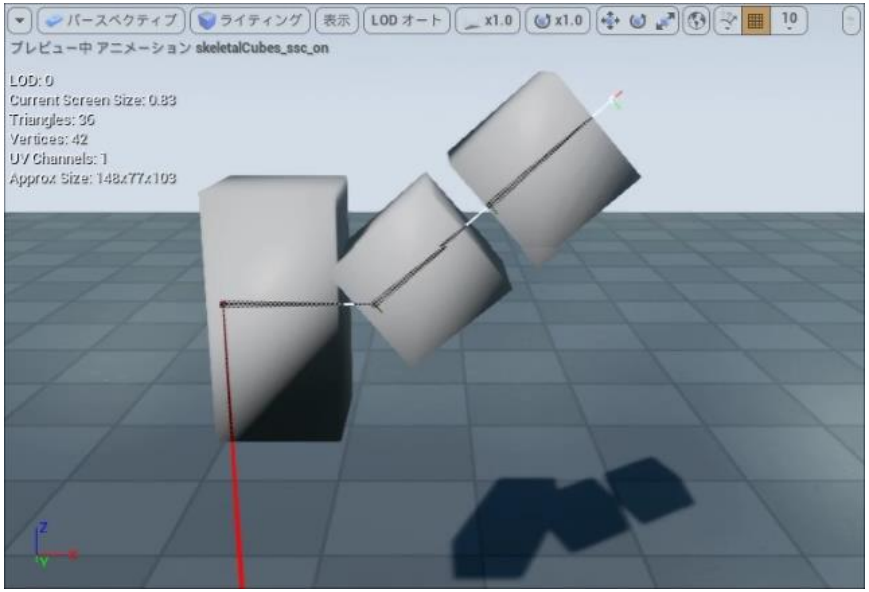

- InheritType を変更した３種のアニメーションを FBX® で UE4 にインポート、 scale アニメーションがどのように再現されるのか調査した。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# セグメントスケール補正 ON の場合



Maya®



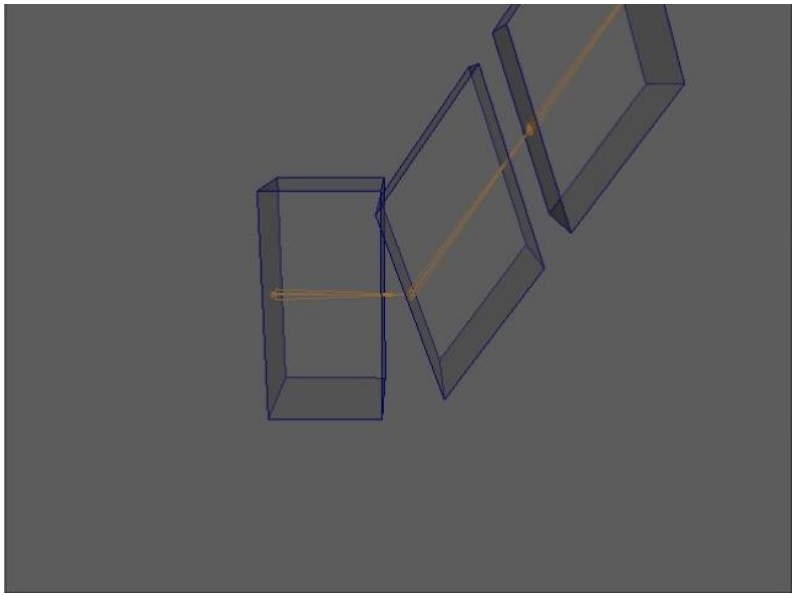

Unreal Engine 4

Join1
  scaleY = 2
Joint2
  ssc = on (InheritType = Compensate)
  rotateZ = 0 ~ 360

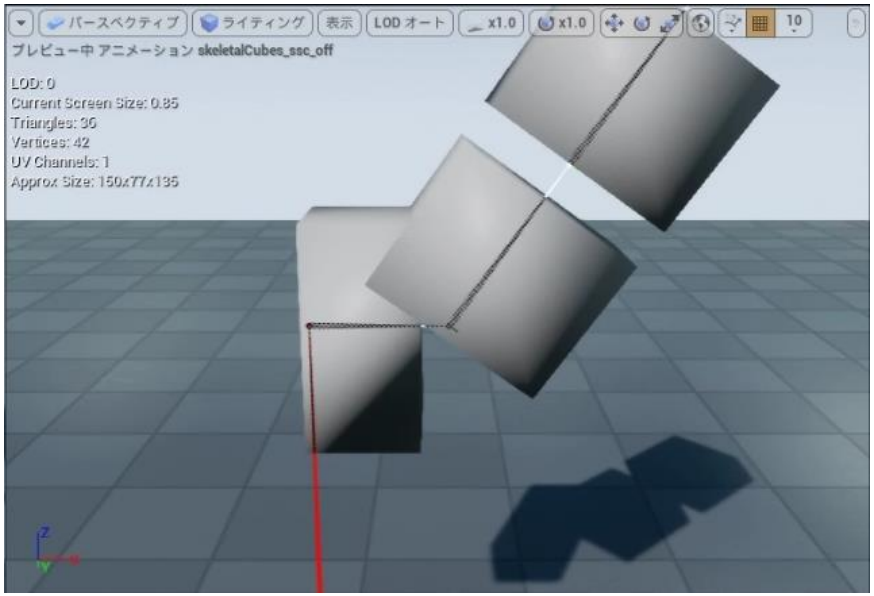
問題なし

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# セグメントスケール補正 OFF の場合



Maya®



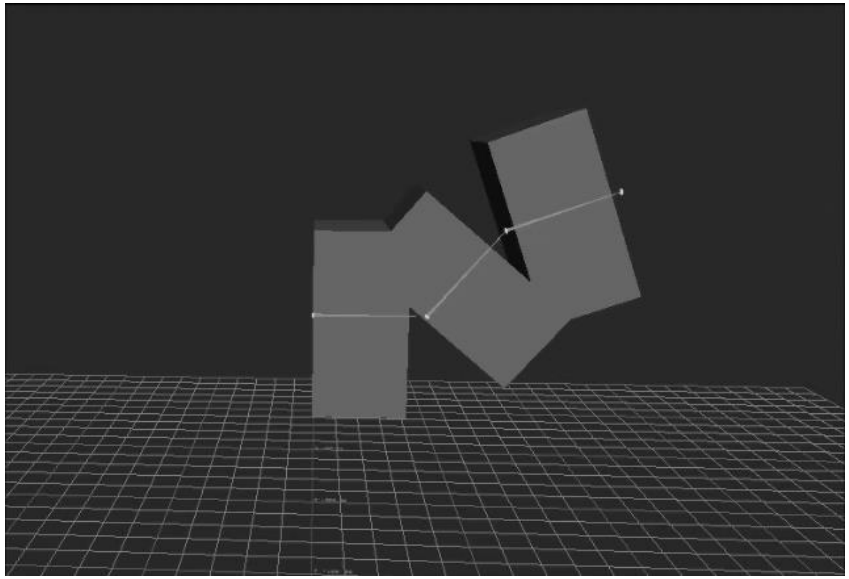

Unreal Engine 4

Join1
  scaleY = 2
Joint2
  ssc = off (InheritType = Parent)
  rotateZ = 0 ~ 360

形が違う
(歪まない)

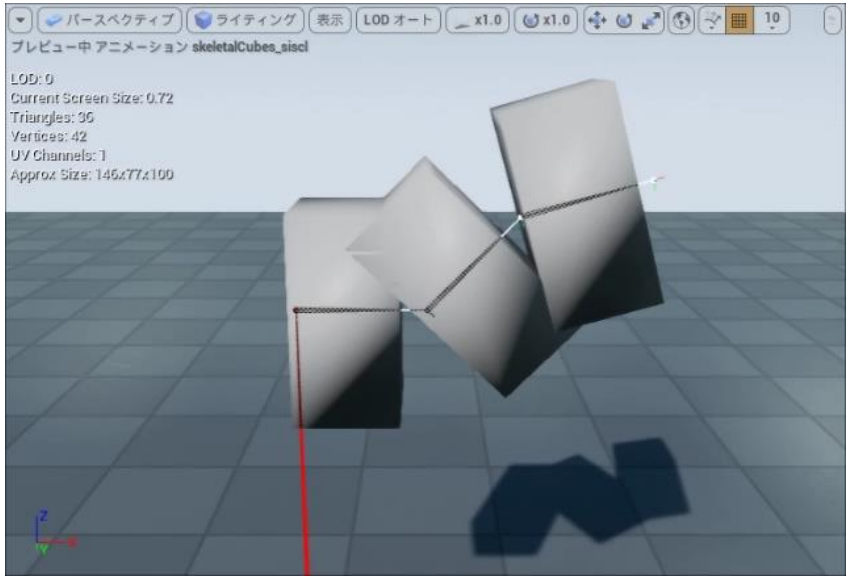
© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# Softiamge® 階層スケーリングの場合



MotionBuilder®

Join1
  scaleY = 2
Joint2
  InheritType = Local
  rotateZ = 0 ~ 360




Unreal Engine 4

問題なし

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# Unreal Engine 4 での scale 再現性

- 同一のスケルタルメッシュでも、アニメーションごとに InheritType を変更可能。**便利。**
- Maya® の ssc=off で不均一スケールが使用されて shear が発生するケースでは同じ形にならない（shear が再現されない）。微妙な違いだと気づきにくいので注意。
- ssc=off でも **均一スケール** なら問題ないため、部分的に ssc=off にして均一スケールを下層に伝搬させることは問題ない。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# ゲームランタイムの一般的な話として

- 各社のゲームランタイムによって scale の対応は様々だろうが、本当は DCC ツールの仕様に左右されるべきではない。
- 単にアニメーションを再生させるだけなら、shear がサポートされているのが単純で良い。
- マトリックス、又は shear も含む分解値 をそのまま再生出来れば、 Maya® のピボットがあーだこーだというトラブルも無い。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# ゲームランタイムの一般的な話として

- 高度なランタイムでは rotate と scale が混ざっているのは都合が悪く、 FBX® のような対応が妥当かも知れない。
- InheritType の違いが完全にサポートされると嬉しい。UE4 は少し惜しい。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# MAYA® でのリギング

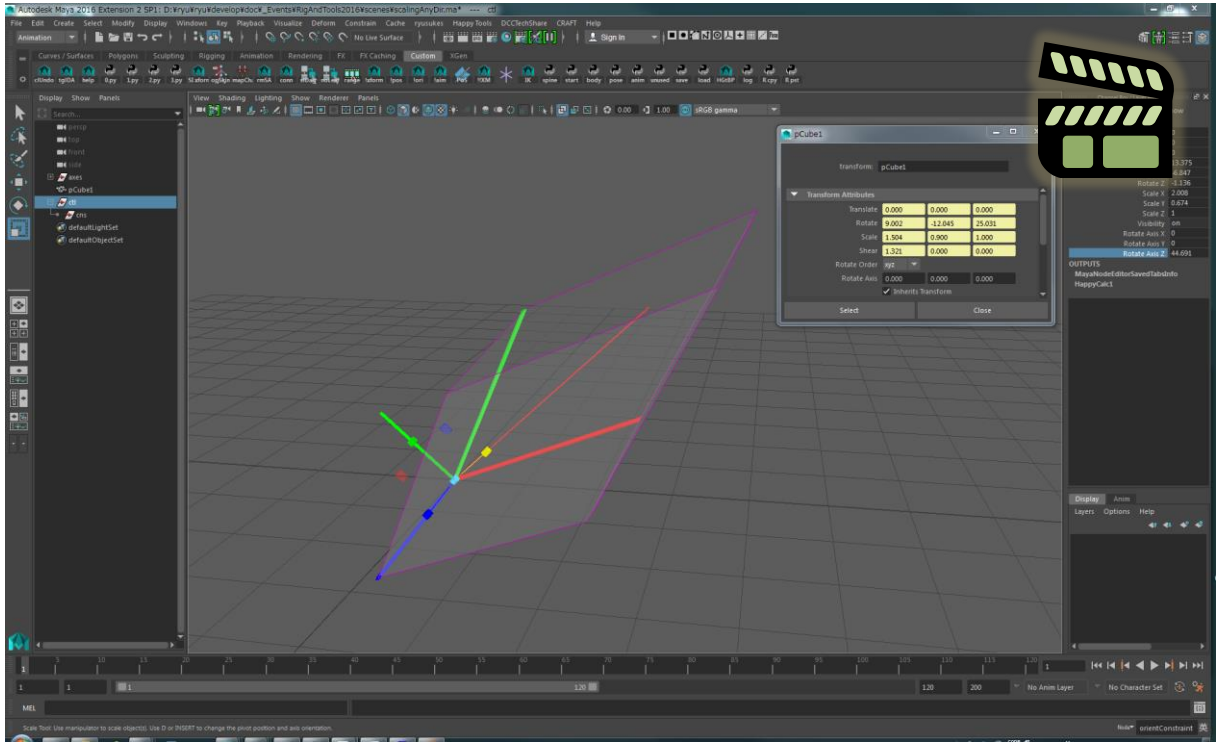
56

## (1) 任意軸でのスケール

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

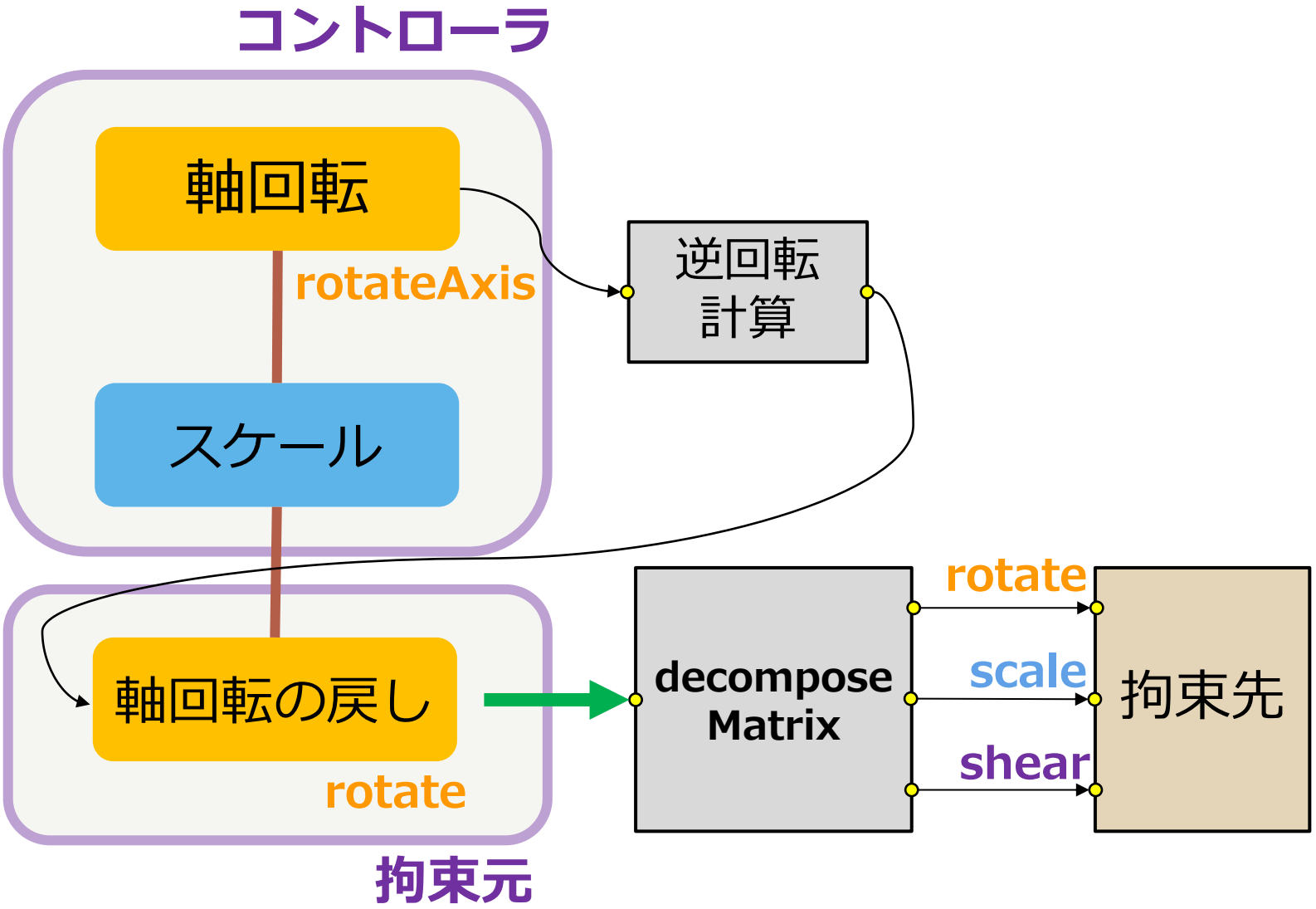
# 任意軸でのスケール

scale のみだと、ローカル軸に沿ったスケールしか出来ないが、rotate, scale, shear を駆使すれば『任意軸でのスケール』が可能。



© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# 任意軸スケールのリグの解説




© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# 逆回転？

- 基本的には、クォータニオン関連のノードが使える。
  - eulerToQuat
  - quatInvert
  - quatToEuler

まったく数学を使わずに逆回転を得る方法

- rotateOrder を逆にする（例えば xyz なら zyx ）。
  rotateAxis のオーダーは xyz 固定なので zyx に。
- rotate X,Y,Z の各値を符号反転する。

マトリックスでの証明：

$$(rx * ry * rz)^{-1} = rz^{-1} * ry^{-1} * rx^{-1}$$

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# 出力先に要注意

- shear は FBX® ファイルに出力されない。
- ゲームランタイムでの shear サポートは？
- 実質、プリレンダームービーでないと無理？


© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# CRAFT のリグモジュール開発で困った

- CRAFT では、スケルトンのジョイントがどんな向きでも問題なくコントロールリグが生成されるようにしていた。
- しかし、scale リグの場合は、ジョイントの向きがローカル軸のどれかに沿っていないと shear が発生してしまう。FBX® やゲームに出力する場合に困る。
- 不本意ながらも、リグ生成時に軸方向をチェックして問題があればエラーにするようにした。また、shear が問題ない場合は、オプションでチェック機構をオフにする。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

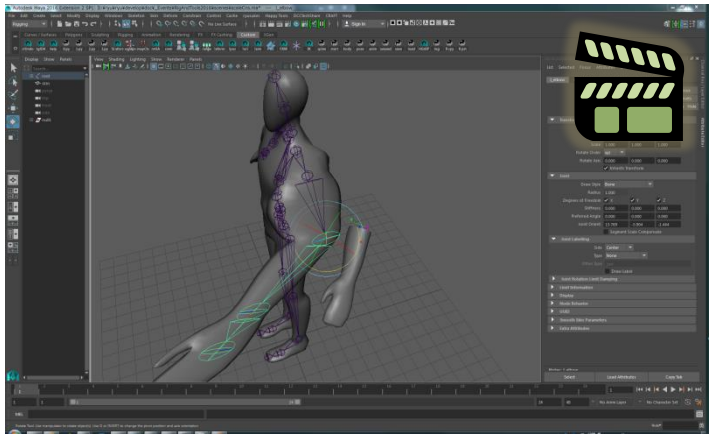
# MAYA® でのリギング

62

## (2) コンストレインについて

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# orientConstraint の問題



- 2013 までは、不均一スケールを親に持つ joint の拘束結果がおかしかった（ssc の考慮がなかったため）。この問題は 2014 で解決された。
- 2016.5 最新版でも、不均一スケールを補正していないソースからの拘束が期待する結果にならない問題がある。ただし、shear をどう捉えるかの尺度の問題ともいえ、不具合とはいえないかもしれない。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# orientConstraint の問題の考察

- 階層上位の shear を無視する実装がされている為。


親のワールドマトリックスから回転を分解
（正規直交化かクォータニオン化）

- shear も考慮するなら、親のワールド回転ではなく
  親のワールドマトリックスを乗じると良い。

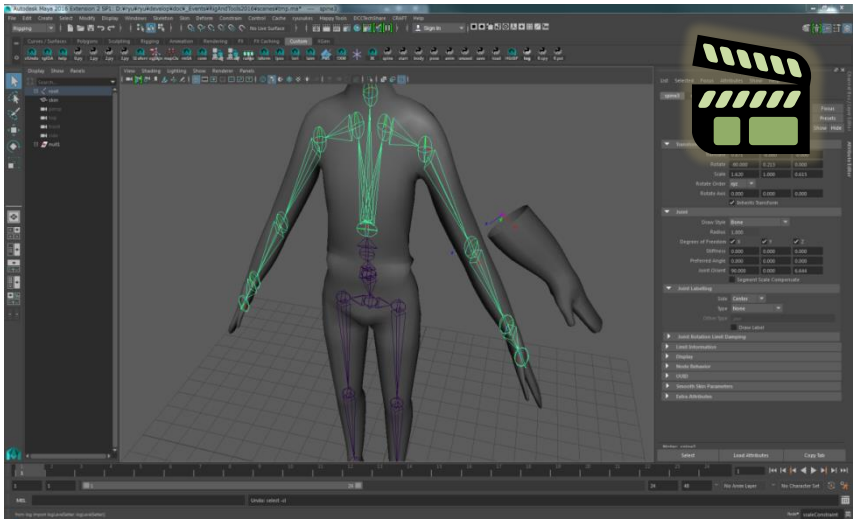

正しい実装


簡易的な実装

どちらも同じ結果が得られるが、上の方が依存が少なくて良い

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# scaleConstraint の問題

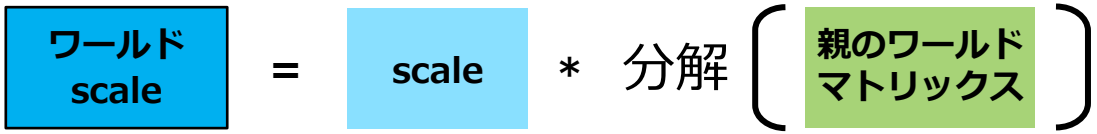


- scale させた joint の直下の子の ssc による scale 打ち消しが考慮されない。
- 親との scale 軸の違いが考慮されないため、ワールド空間 scale を得たところで何だか分からない。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# scaleConstraint の問題の考察

- 拘束元ノードの rotate（親とのスケール軸の違い）を考慮しない実装。shear も無視。




親のワールドマトリックスから
scale を分解

- 親との軸違いを考慮するためには、拘束元ノードの rotate も含まれたワールドマトリックスを処理する。


decomposeMatrix
ノードが使える

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

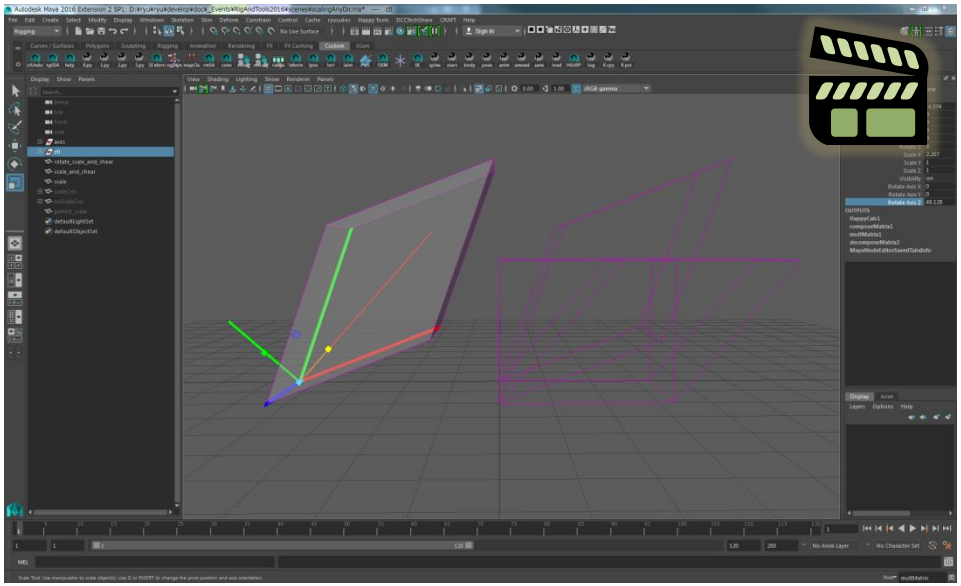
# コンストレインの組み合わせの問題

- 例えば、回転とスケールを完全に拘束したい時、orientConstraint と scaleConstraint を組み合わせても、拘束元と同じ結果にならないことが多い。
- 不均一スケールが混在した場合、rotate と scale と shear は密接な関係にあり、個別には扱いにくい のが原因。
- 標準機能のコンストレインは、rotate と scale を個別に扱おうとするあまり相互影響を無視しているため、見た目期待する結果にならないことが多い。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# 問題解決のために

- decomposeMatrix ノードを使うだけでもかなり解決出来る。
- rotate と scale を独立させたような制御ギミックを作りたい場合、コンストレイン機能任せではなくノードの組み合わせを自分で組む方が良い。



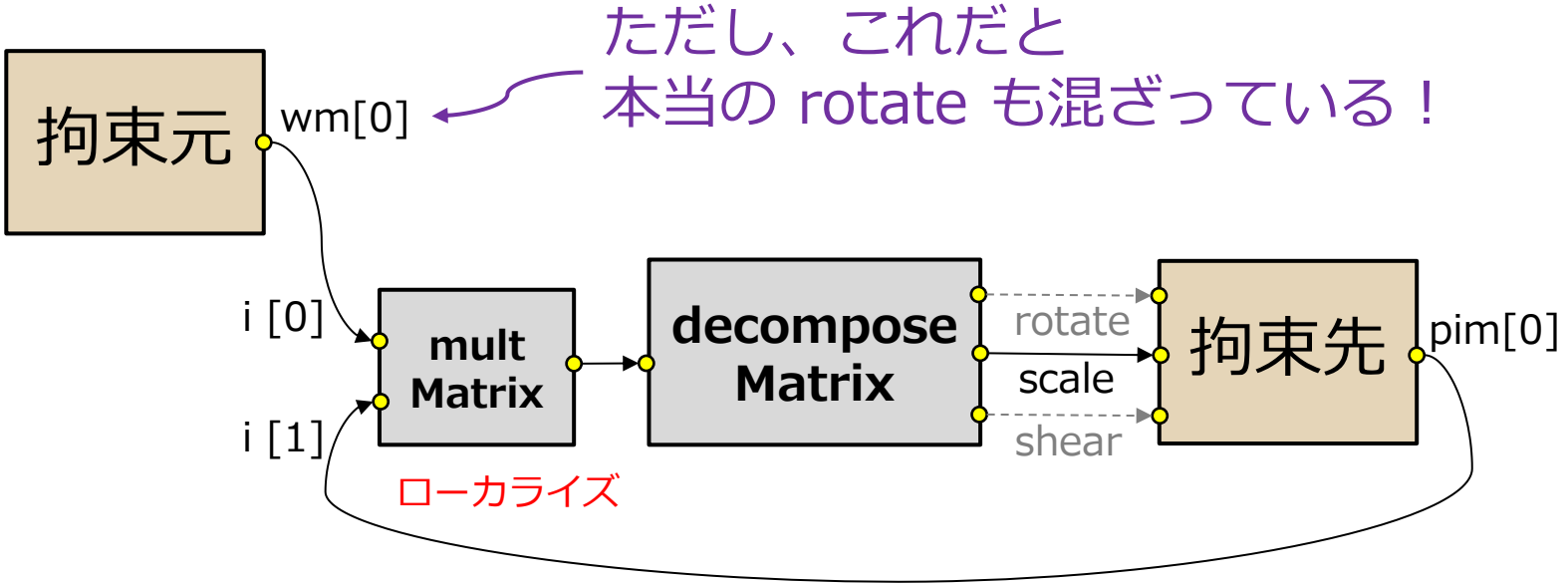
© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# shear を伴う scale 制御のコツ

- 基本的には scale と shear はセットで扱う。
  shear は不均一スケールと rotate の利用とともに意図せず発生する scale 要素。
- さらに、scale & shear によって発生する rotate 成分もあれば、本当の rotate とは混ぜずに scale とセットで扱うとより良い。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

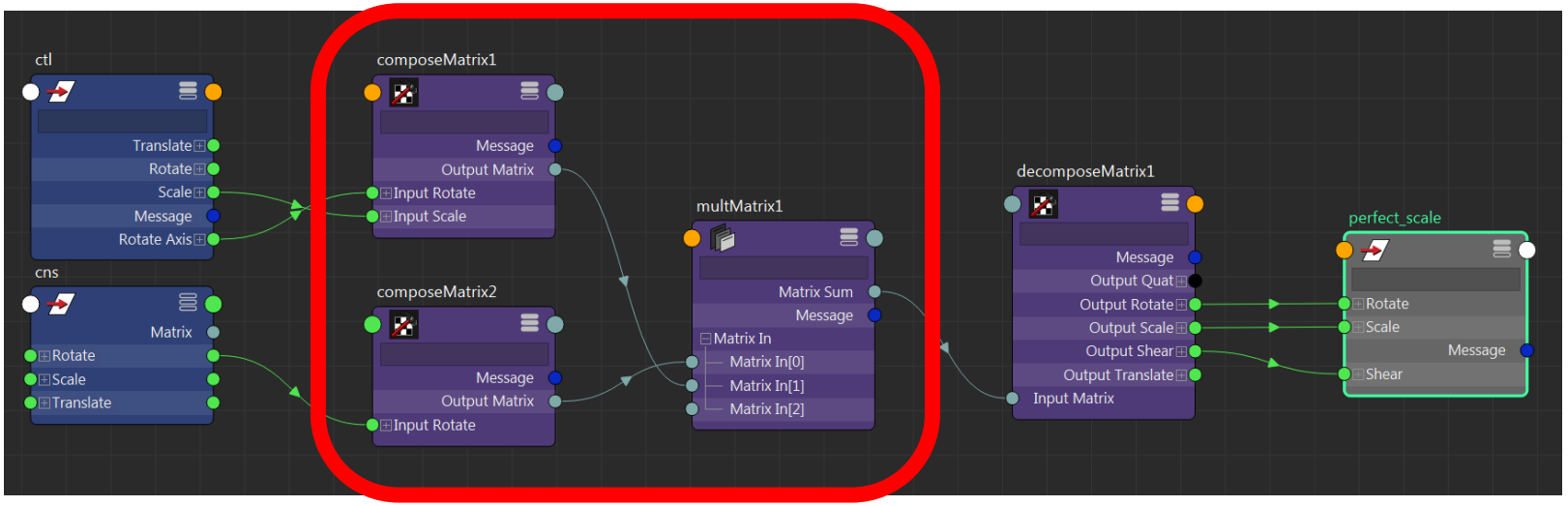
# scale コンストレインの組み方の例




- multMatrix ノードを使って、拘束先の親空間でローカライズする。
- 必要に応じて shear も拘束する。
- さらに、必要に応じて rotate も拘束する。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# 本来の rotate を混ぜない拘束




「軸回転」「スケール」「軸回転の逆」の３つを合成してマトリックスを生成して decomposeMatrix する。

worldMatrix を評価しないローカルリギングの例。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# MAYA® でのリギング

72

## (3) コントロールリグでも joint を使う

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# リグでは基本的に shear を避ける

- このように、shear が入り込むと大変ややこしいことになるので、基本的に shear を避けるリグを組むのが良い。
- shear を避けるには、不均一スケールと rotate を混ぜないこと。
- そのためには ssc（セグメントスケール補正）。リグのコントローラでも joint ノードを使って、入力された不均一スケールが、その下層に影響しないようにする。
- そもそも shear アトリビュートも使わないように。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# transform ノードの代わりに joint を使う

- joint タイプは transform タイプの派生型なので、普通に transform ノードの代わりとして使うことが出来る。
- 直接 shape ノードを持つことも出来る。
  joint の機能を持った locator を作ってコントローラにすることが出来る。
- もっと凝る場合は、プラグインで独自の transform ノード型を作ることも出来るが、私はうまくいかなかったので諦めた経験がある。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# joint に shape を持たせる方法

hoge1 という名の joint locator を作る例。

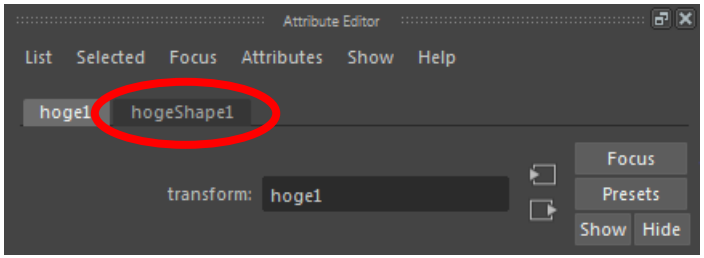
```
import maya.cmds as cmds

j = cmds.createNode('joint', n=r'hoge#')
cmds.createNode('locator', p=j, n=r'hogeShape#')
cmds.setAttr(j + '.drawStyle', 2)
```

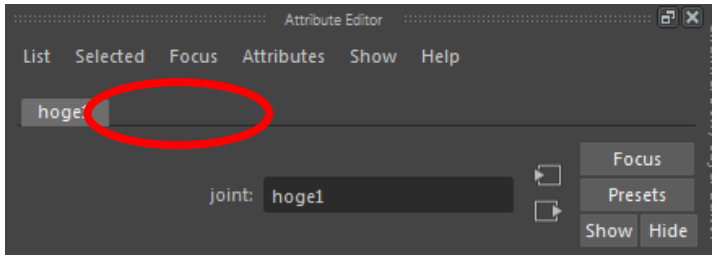
- createNode コマンドで parent 指定するだけ。
- ついでに drawStyle に 2 (None) をセットして、joint が描画されないようにしている。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# AEjointRelated.mel

- Attribute Editor で shape にアクセスしにくいというほんの些細な問題がある。




- 以下の内容のMELスクリプトを AEjointRelated.mel というファイル名で置いておくだけで解決。

```
global proc string[] AEjointRelated(string $nodeName)
{
    return AEtransformRelated($nodeName);
}
```

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# joint のメリット

- jointOrient が使える。
  - translate の軸と rotate の軸を変えることが出来る。
  - コントロールノード数を減らしてアクセスしやすく出来る。使いやすいリグを作る上では重要。
- ssc（セグメントスケール補正）が使える。
  - scale リグではとても重要！
  - 実は、inverseScale には好きなものを繋ぐことも出来る。通常は親 joint の scale を繋ぐものだが、独自の繋ぎ方をしてより凝った scale リグを作れる。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

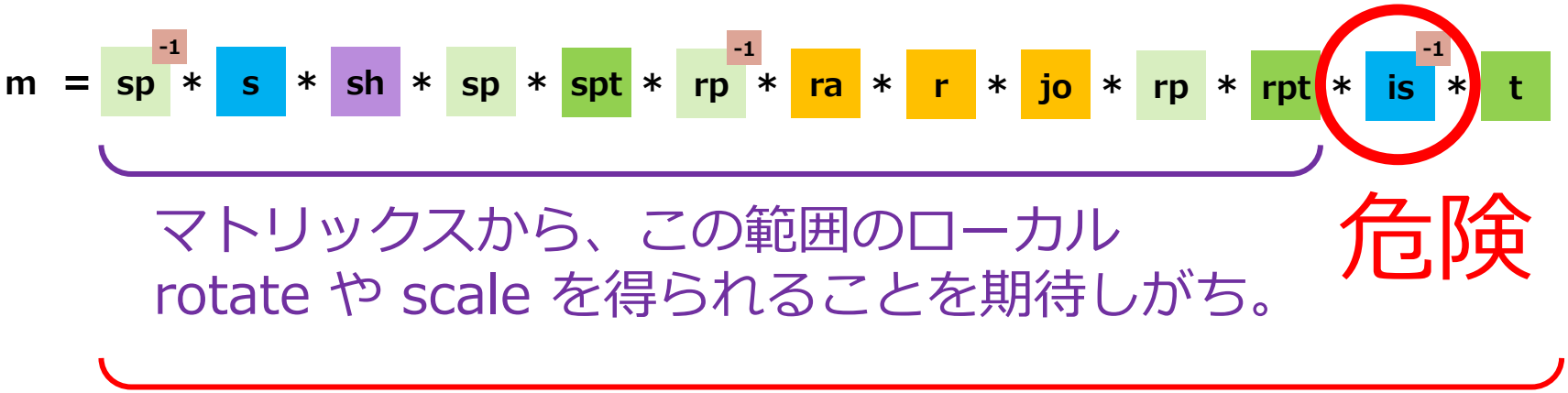
# joint のデメリット

- ピボットを変更できない。
- 移動コントローラとして使った場合、Move Tool の jointOrient 自動設定機能が有効だとうるさい。
  - 親も joint の場合に、この機構が発動する。
  - 親子とも rotate がゼロじゃないと警告メッセージがうるさい。
  - 移動可能な joint コントローラの親を joint にすることは避けると良い。
- drawStyle を None にすると displayLocalAxis での軸表示もされなくなる。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# ローカル空間リギングにおける注意点

joint のローカルマトリックスには inverseScale が含まれている。気をつけないと本当にハマる。




ローカルマトリックスには親のスケールを打ち消すための逆スケールが含まれているので、歪んだマトリックスとなる。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# SSC に過信は禁物

- 打ち消すのは、通常は 親のローカル scale である。
  それより上位の歪みが取りきれていないと困る。
  リグでは、inverseScale に独自のコネクションを
  作れるので何とかなる。
- inverseShear は無いので shear は打ち消せない。
  そもそも shear を使わないようにすれば大丈夫。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# MAYA® でのリギング

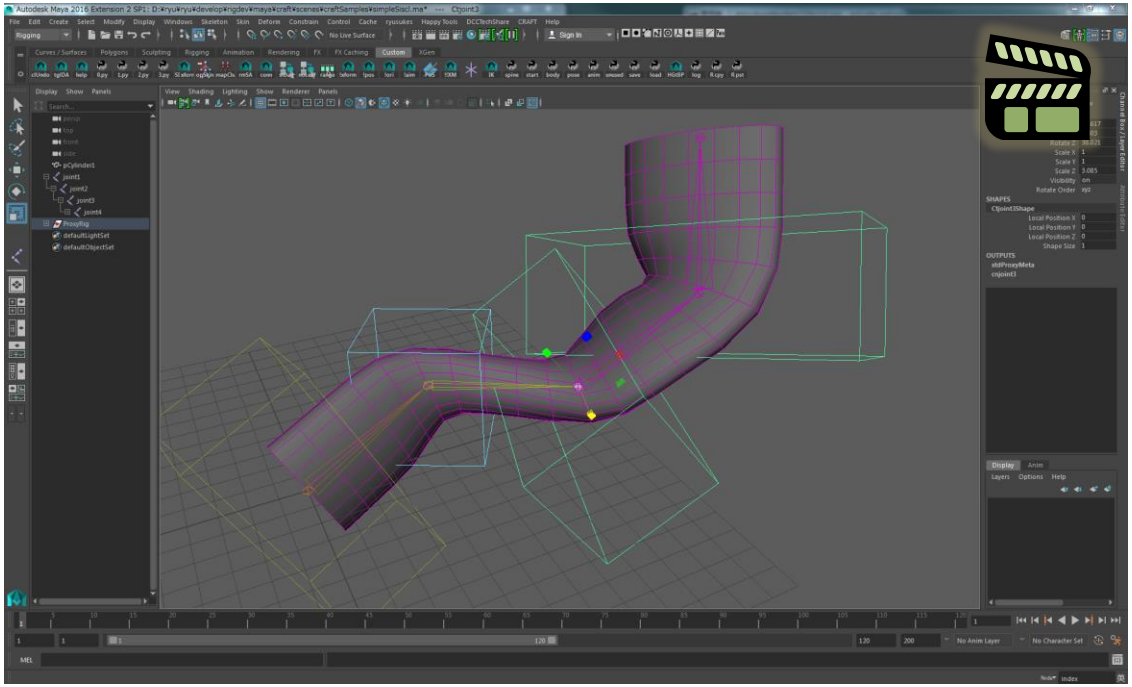
81

## (4) Softimage® 階層スケーリングをリグで使う

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# Softiamge® スケーリングが使いたくなる

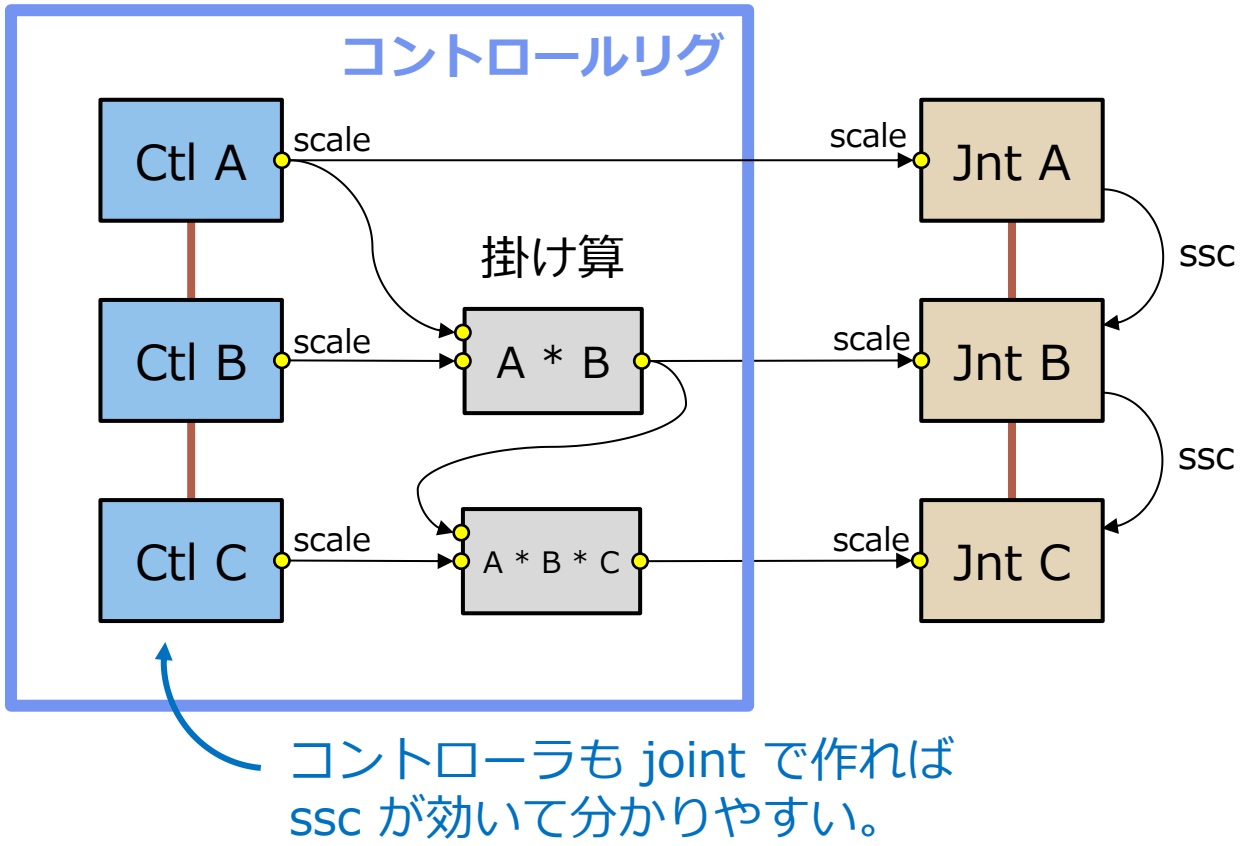
- ssc は便利だが、コントローラとしては子に scale を伝搬させたいことが多い。
- まさに Softimage® スケーリング的。



© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# ssc を使えば基本的な考え方は簡単

- rotate を歪ませない仕組みとして ssc は使える。
  その上で、階層下の scale もリグが面倒見れば良い。




© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

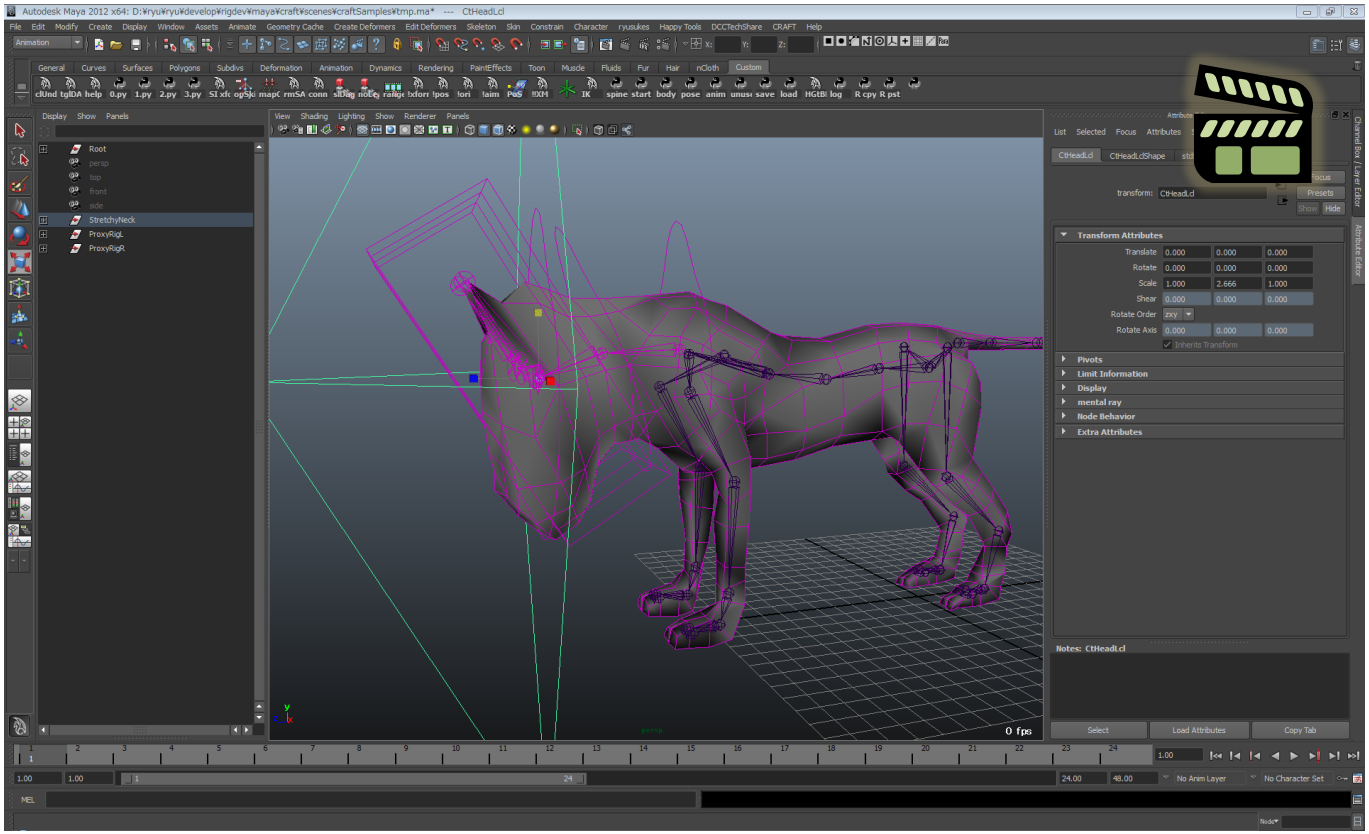
# 補足: Maya® は汎用ノードが少ない

- 例えば、ごく簡単な「掛け算」といっても…
  - multDoubleLinear
    - 単位が distance 型。scale とは型が違う。
    - 3 値の乗算には 3 ノード必要。
  - multiplyDivide
    - float3 (x, y, z) の乗算や除算が出来る。
    - 単位が float なので精度が悪い。問題なければOK。
  - プラグインノードを作る
    - 可能ならベスト。
  - expression
    - プラグインノードにしない場合は、これが最適か。
    - デフォルト設定ではなく、殆ど常に以下の設定で使うこと！！
      - Convert Units: None
      - Evaluation: On demand

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# 自動的な scale 軸合わせ

- 対応させる scale 軸を、最も近い者同士に合わせるように組むと良い場合がある。



© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# まとめ



© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# まとめ

- scale が難しいのは shear のせい。不均一スケールさせた下の rotate で発生し、色々と問題が起こる。
- 各ツールは shear を避ける工夫をしていたり、そもそも考慮していなかったりする。
- 少なくとも、shear を使えば、異なるリグ空間やツール環境で同じ姿勢を再現することは出来る。
- とはいえ、rotate, scale, shear は密接した関係にあり、一旦混ざってしまうと個別制御が難しい。
- 結局、リギングでは rotate と scale を混ぜない組み方をして shear を避けるのが望ましい。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

# 本音のまとめ

- それでも… scale リグは難しい。
  もうマジ勘弁。
- ゲームランタイムも然り。不均一スケールを使用しないに越したことはないです。
- アニメータの皆さん、あまり、スケール、スケール言わないでください…。

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.

ご清聴ありがとうございました。

何かご質問はありますか？

掲載されている会社名、商品名は、各社の商標または登録商標です。

佐々木隆典 ryusukes@square-enix.com

SQUARE ENIX®

© 2016 SQUARE ENIX CO., LTD. All Rights Reserved.