次世代を見据えた
新しい補助骨システムの開発

株式会社スクウェア・エニックス
テクノロジー推進部
佐々木　隆典
髙木　啓太

SQUARE ENIX ADVANCED TECHNOLOGY DIVISION

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

登壇者紹介

佐々木 隆典

- テクノロジー推進部所属 テクニカルアーティスト（プログラマー）
- 補助骨システム KineDriver 初版の設計・開発を担当
- 現バージョンでは「補助骨自動生成」などの付加機能を担当

高木 啓太

- テクノロジー推進部所属 テクニカルアーティスト（プログラマー）
- Maya® 版以降の開発の主担当
- Unreal® Engine 4 版の開発も担当

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

アジェンダ

- システムの概要
- Maya® でのオーサリング
- 中間ファイルについて
- Unreal® Engine 4 プラグイン
- 諸機能の実装
  - エクスプレッション
  - RBF 補間
  - 補助骨自動生成
- まとめ

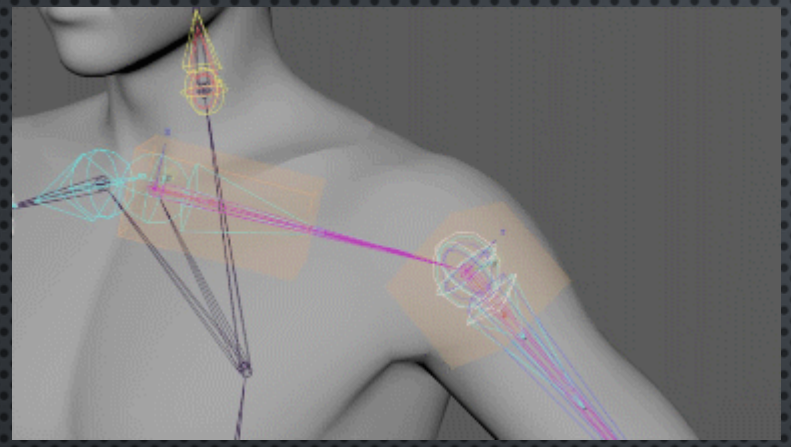
© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

システムの概要

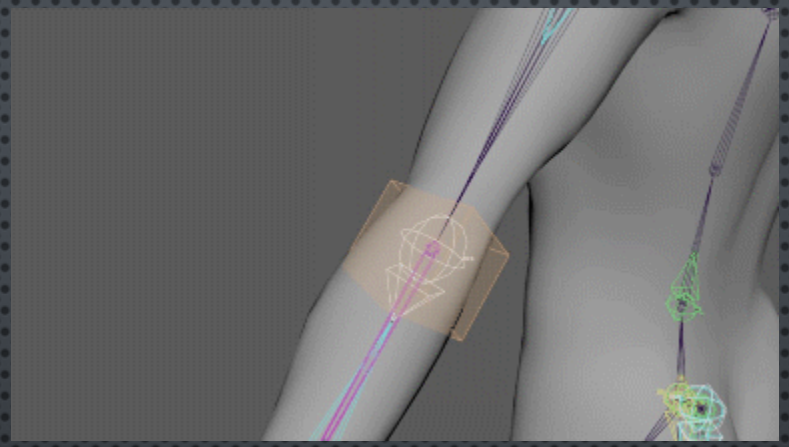
補助骨システムとは

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

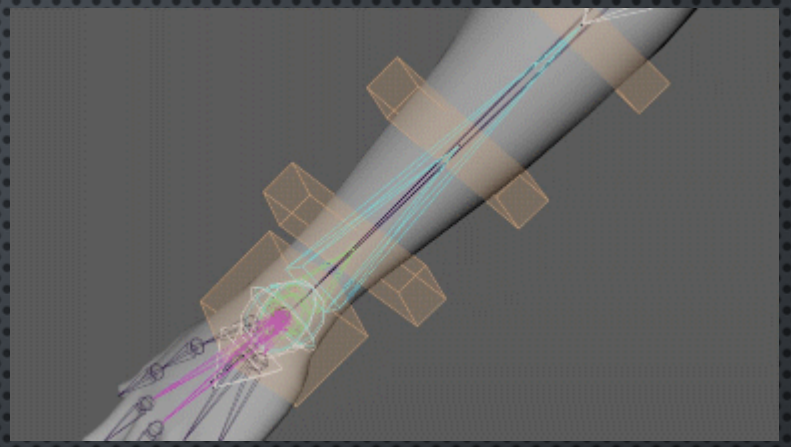
補助骨とは



肩周辺の例



肘の例



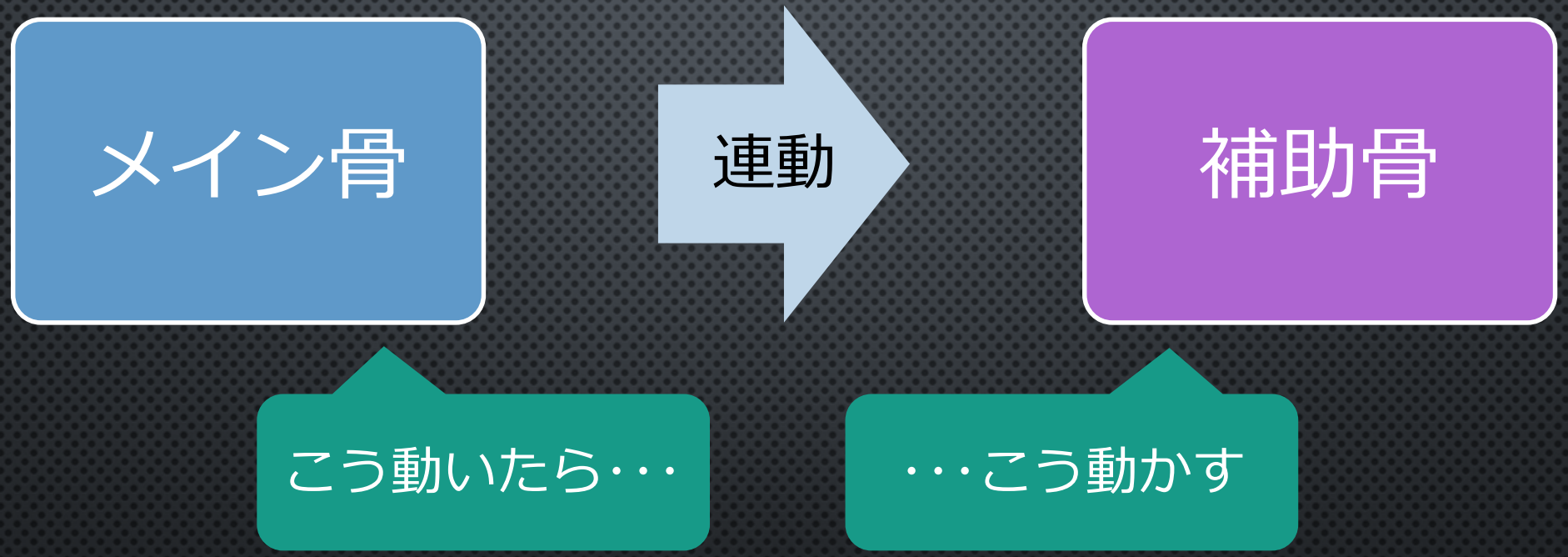
手首の例

主にスキン・デフォーメーションを補う補助的なボーン。

リアリティを向上させることを目的とするが、必ずしも実際の骨格に基づくわけではない。

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

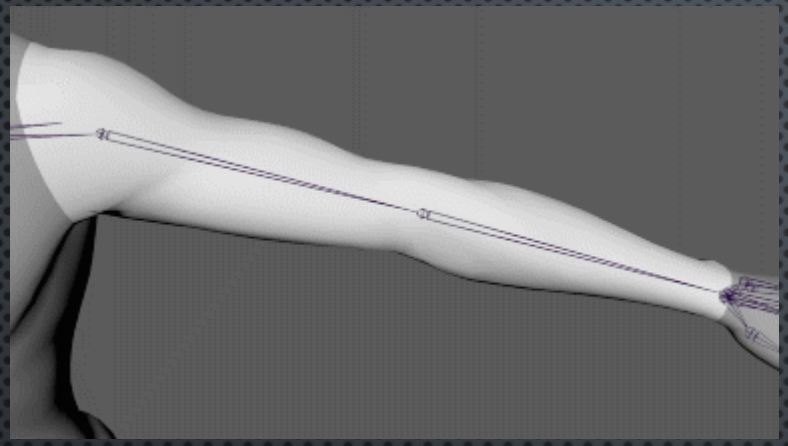
補助骨の動きの制御



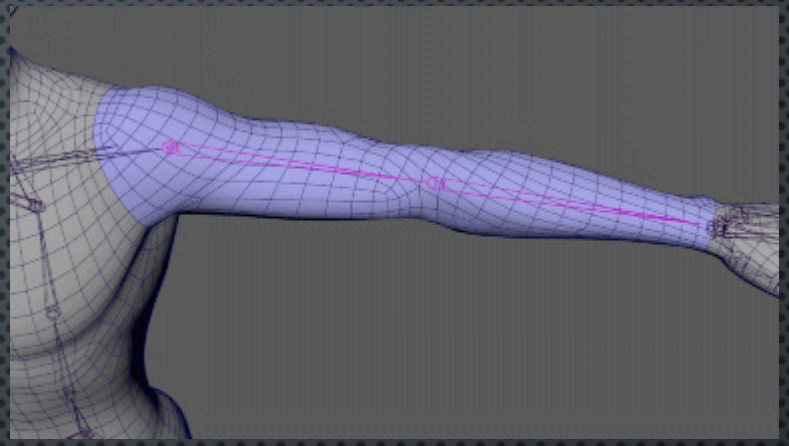
補助骨は、メインの骨に連動させて自動的に動かすのが理想。

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

補助骨のしくみの応用



シェーダー制御

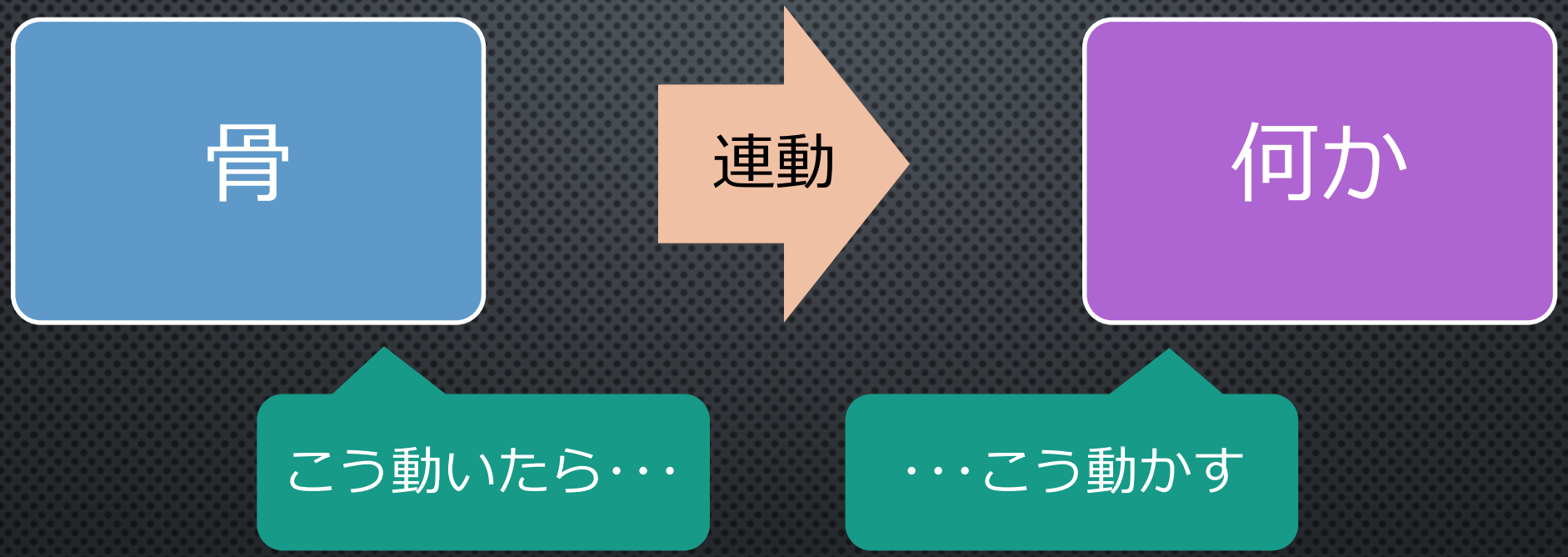


Pose Space Deformation

骨の動きに応じて別の「何か」が動くという仕組みは、
様々な機能として応用することができる。

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

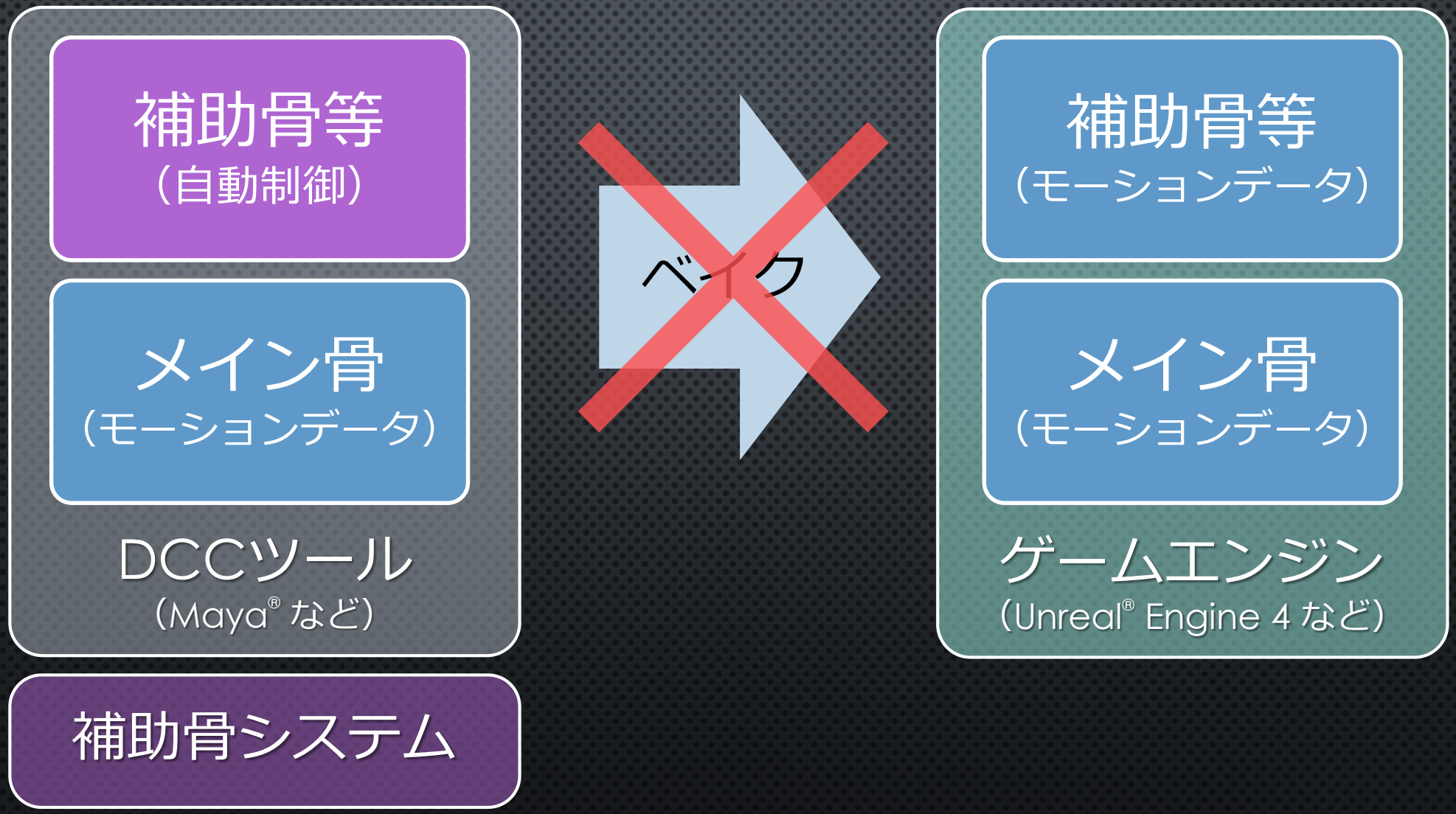
補助骨システムの根幹



骨の動きに応じて、別の何か（補助骨や様々なパラメータ）を動かす仕組みとなる。

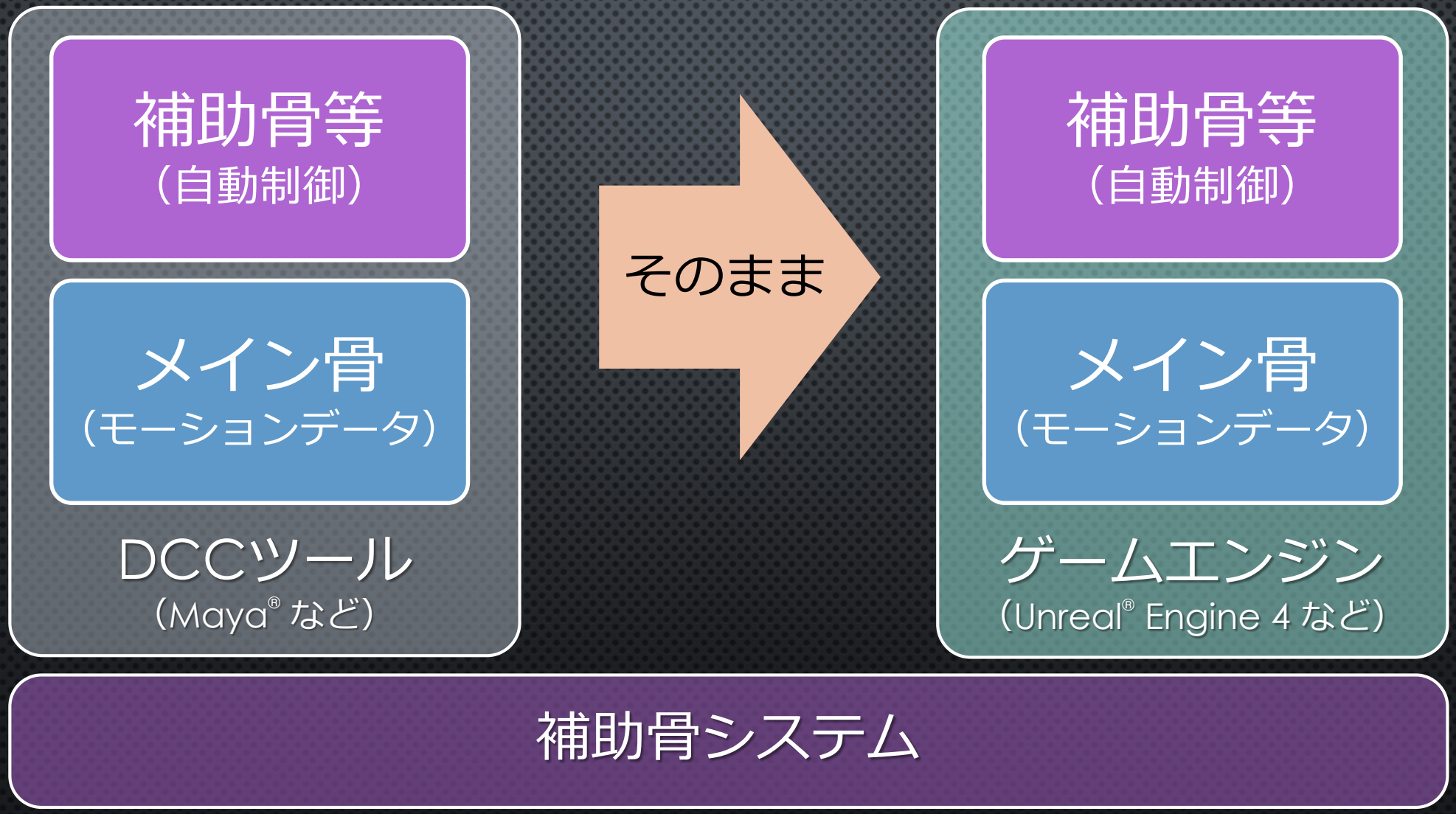
© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

ゲームランタイム



© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

ゲームランタイム



© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

ランタイム補助骨システムの意義

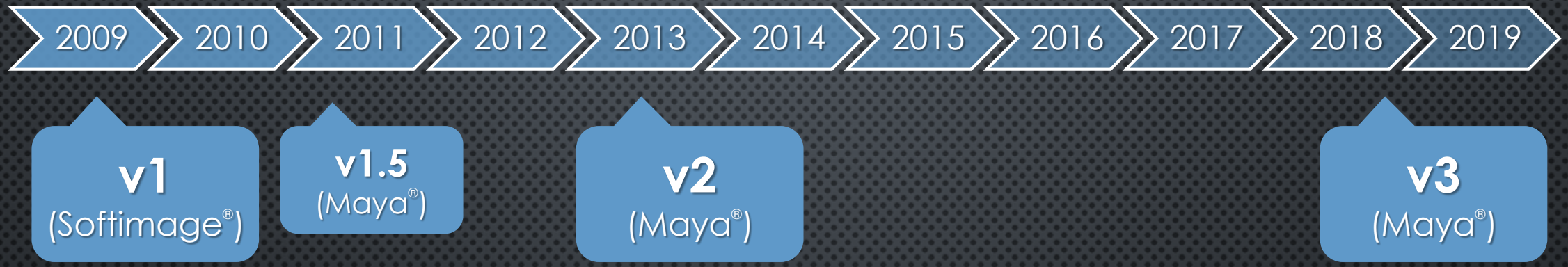
オーサリング環境（DCCツール）と
互換性のあるランタイム補助骨システム

モーションデータ
量の削減

ランタイムでの
アニメーション補正や
生成に対する反応

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

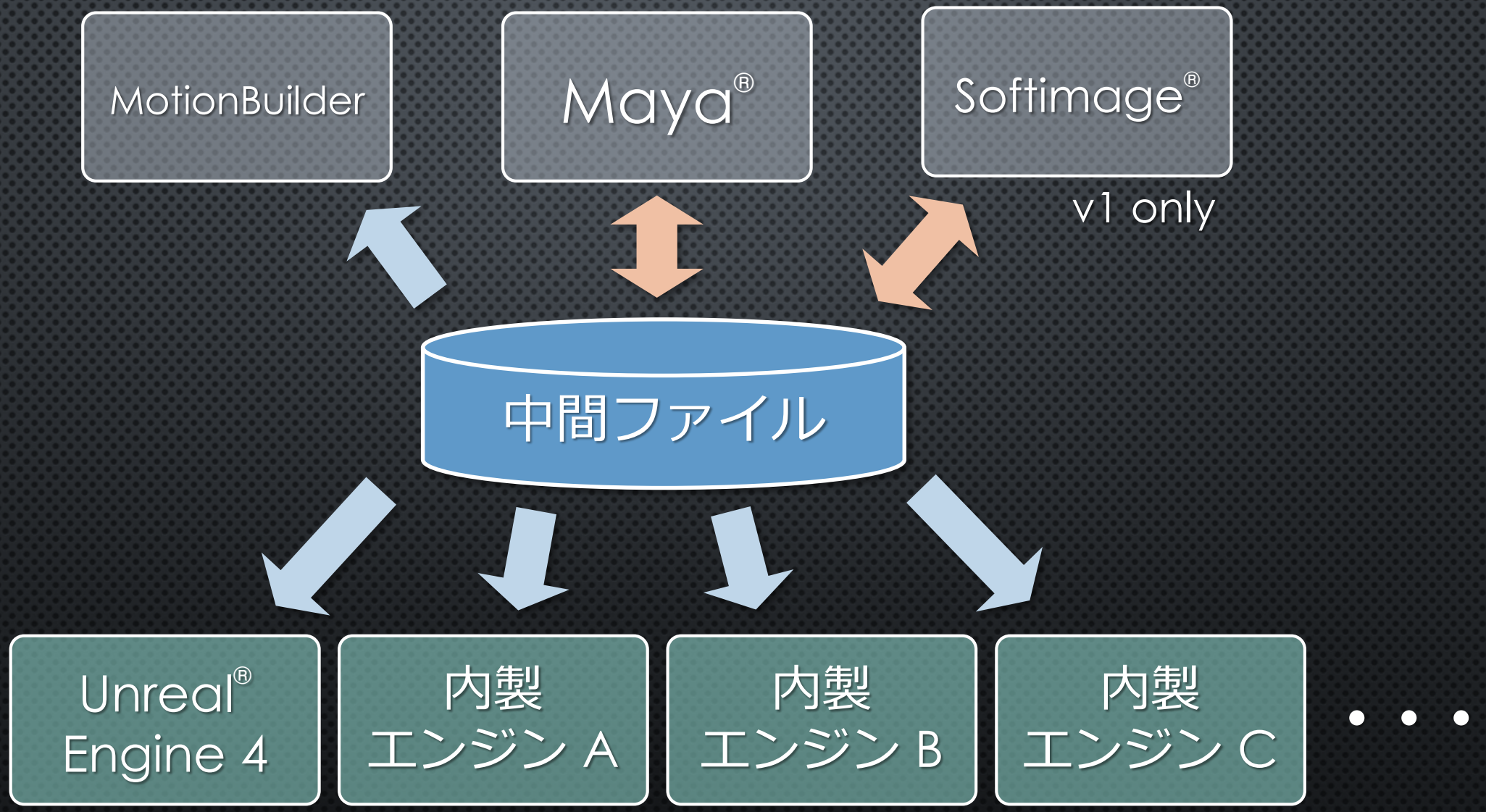
補助骨システム KineDriver



- **v1** … Softimage® をオーサリング環境とする。
  - **v1.5** … Softimage® 互換の Maya® 版。
- **v2** … Maya® をオーサリング環境とする改良版。
- **v3** … 自由なノードグラフの導入。いくつもの新機能追加。

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

サポート環境



© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

システムの概要

機能と基本概念

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

機能分類

Source オペレータと Target オペレータ

- 複数ソースのブレンド
- 基準空間の指定
- 扱いやすい回転表現
- 扱いやすいスケール表現
- エフェクターノードグラフ

Constraint オペレータ

- Position, Orientation, Direction
- メッシュアタッチ機能

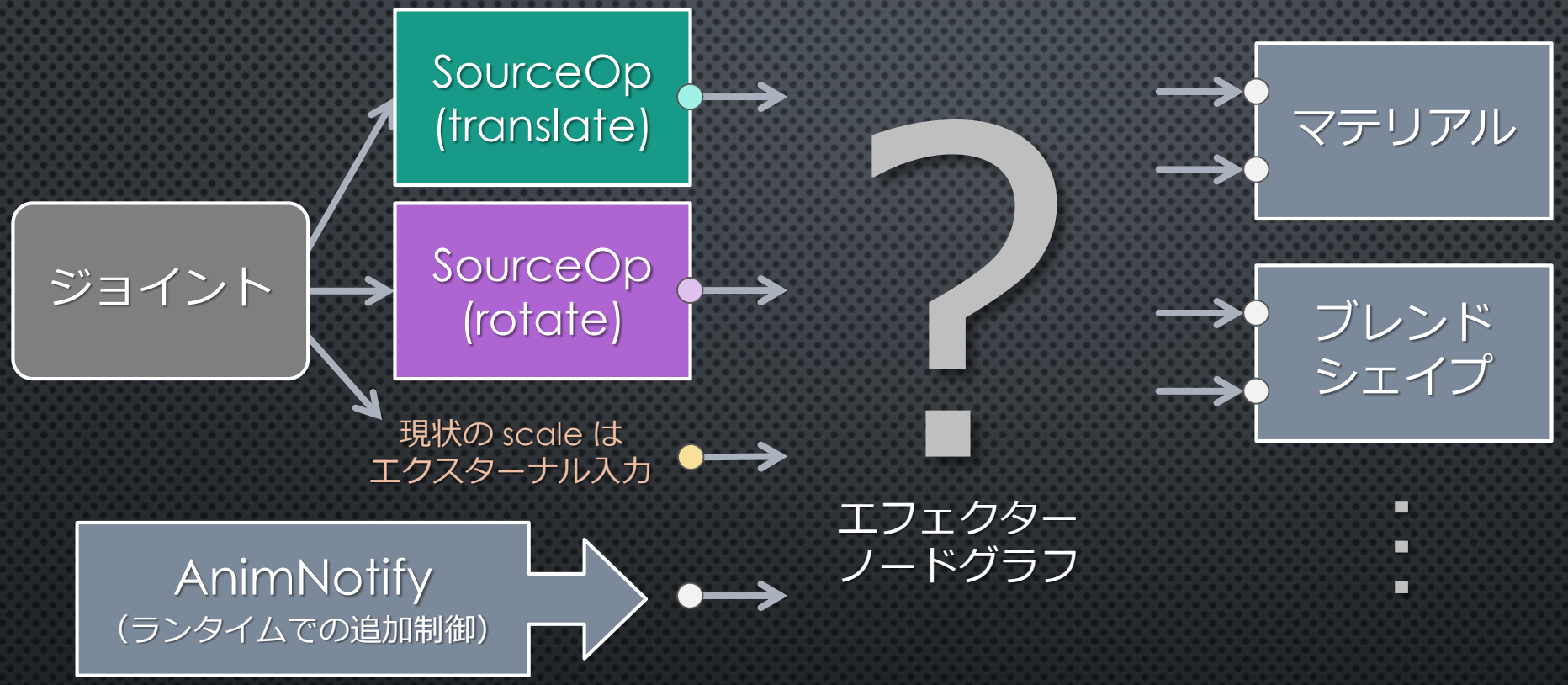
© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

Source オペレータと Target オペレータ

ソース骨には Source オペレータ、補助骨には Target オペレータを取り付けることで、ノードグラフに参加させられる。

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

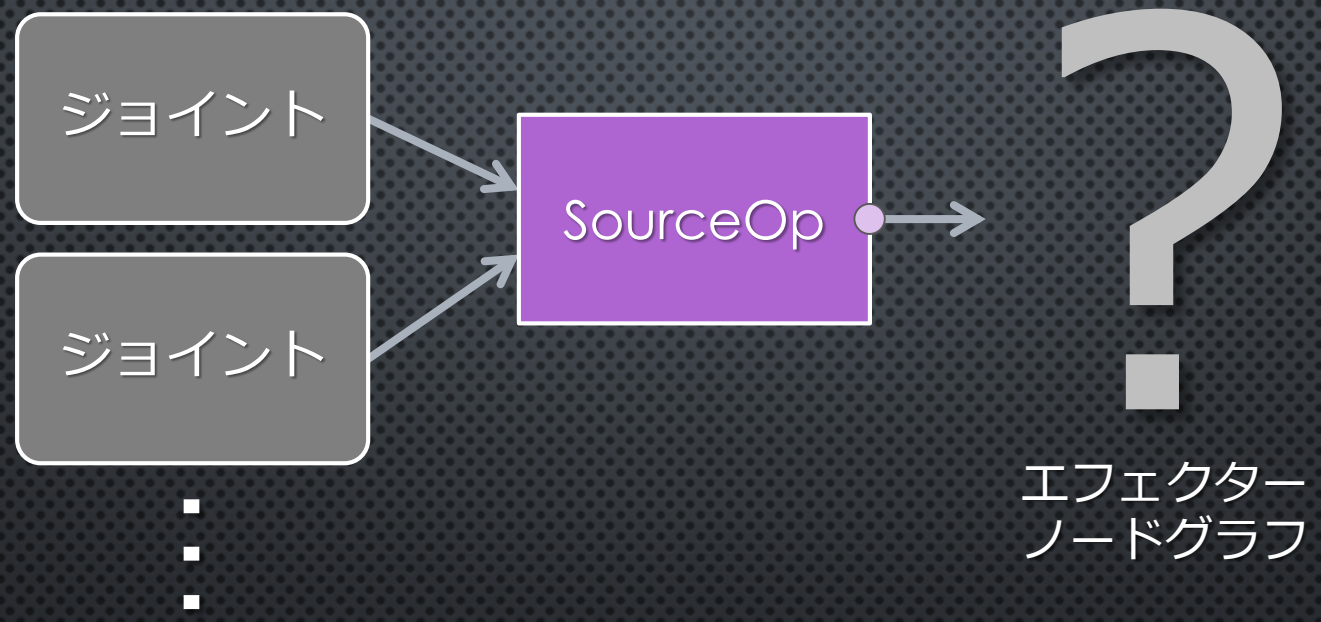
エクスターナル入出力



エンジン側の対応しだいで、骨ではない様々なパラメータ制御の入出力も統合できる。

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

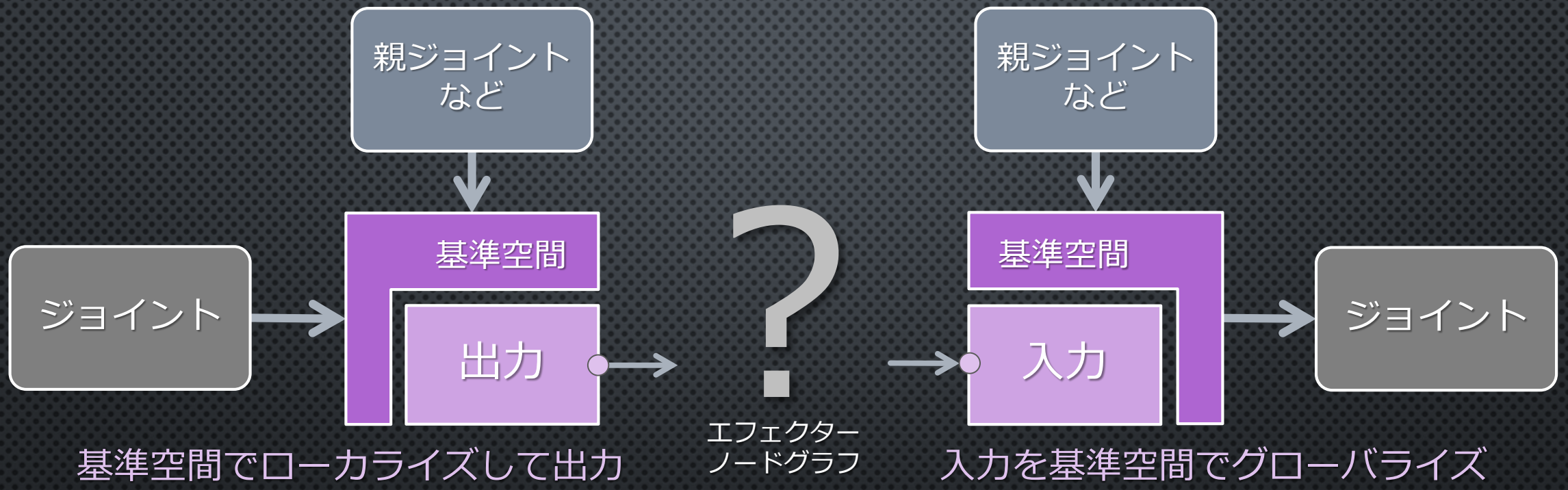
複数ソースのブレンド



Source オペレータは、複数の関節の位置や回転のブレンド機能を備える。

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

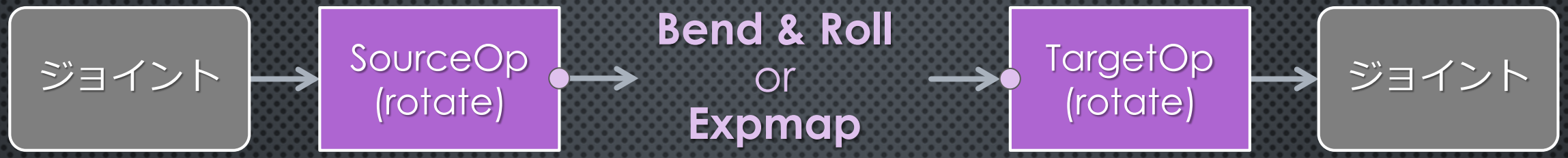
基準空間の指定



デフォルトでは、それぞれの親ジョイントが「基準空間」だが、任意のジョイントを指定して、それらを基準としたローカルトランスフォームの入出力ができる。

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

扱いやすい回転表現



Bend & Roll （曲げとひねり）

- 骨の回転を「曲げ（縦横２方向）」と「ひねり」に分解して扱う。
- アーティストが理解しやすいため、通常はこちらが用いられる。

Expmap（Exponential Map)

- クォータニオンの対数写像。
- 後ほど説明する「補助骨自動生成機能」では、こちらが用いられる。

どちらの表現形式も、オイラー角のような３つの角度の組み合わせ

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

オイラー角の問題点

回転のアニメーション表現によく用いられる「オイラー角」は、補助骨の自動制御には適さない。

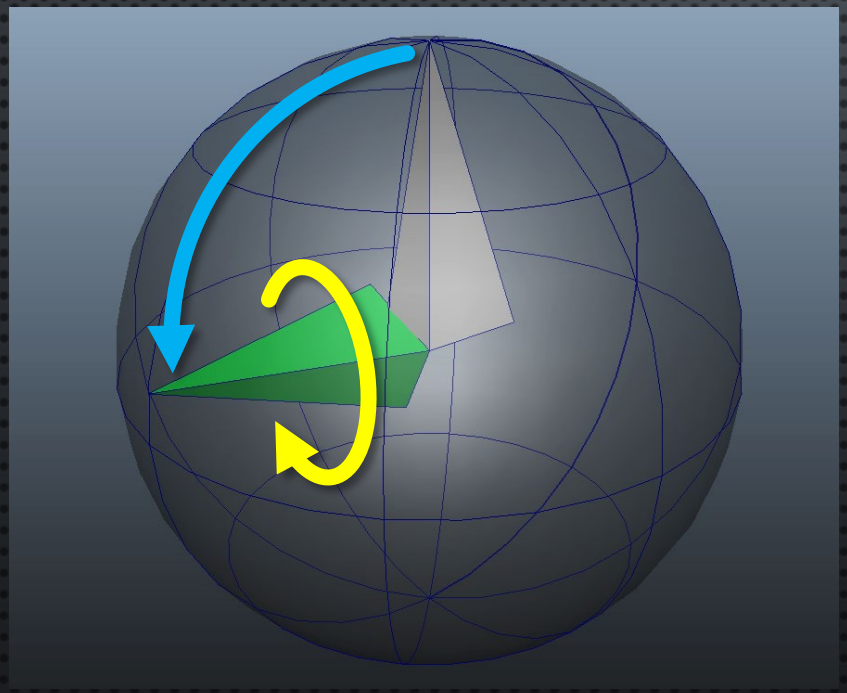
- そもそも、回転の計算が困難。
- 一つの姿勢を表す「値の組み合わせ」が一意ではない。
  たとえば (0, 90, 60) と (-30, 90, 30) は同じ姿勢。
- ランタイムでは、クォータニオンやマトリックスで表現される。
  オイラー角に変換するにしても、期待される「値の組み合わせ」を再現できる保証はない。

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

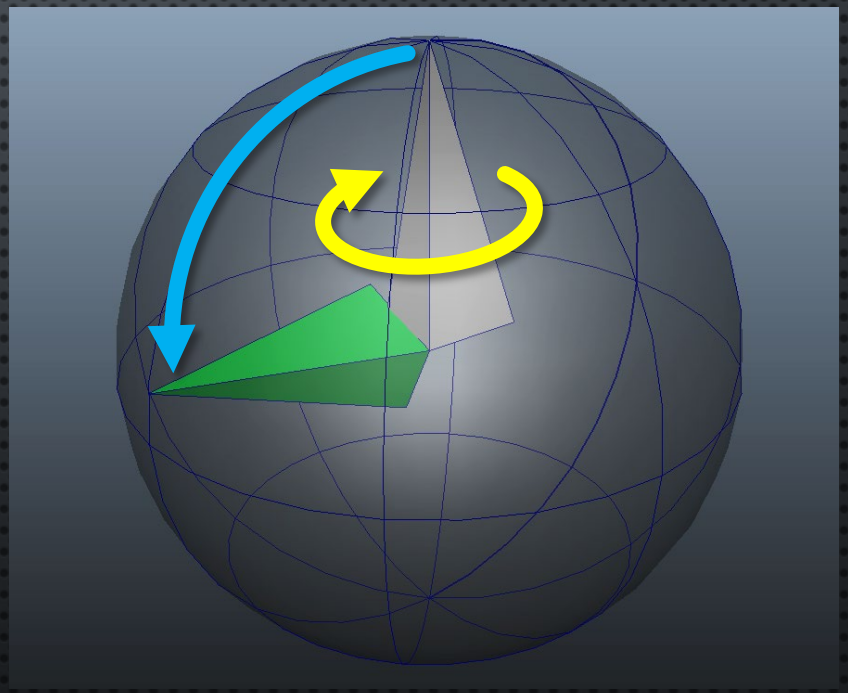
Bend & Roll （曲げとひねり）

曲げ　ボーンベクトルが目標方向に向くための最短回転。

ひねり　目標姿勢から「曲げ」を差し引いた残りの回転。



曲げてからひねり



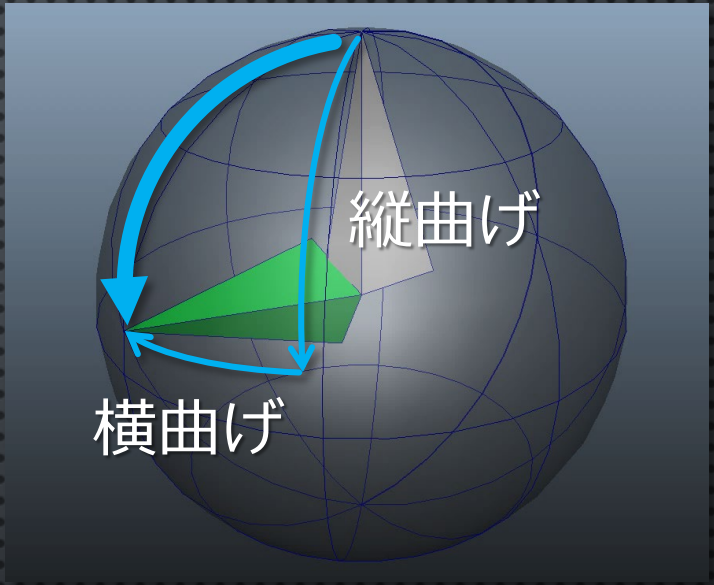
ひねってから曲げ

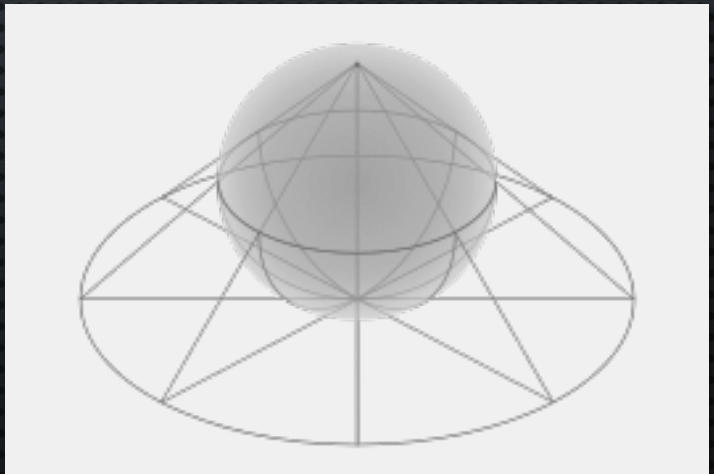
© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

曲げ方向の分解

「ステレオ投影」という古典的手法を用いて、
「曲げ」を縦横の２方向の回転角度に分解。

Stereographic projection (July 15, 2019, 19:02 UTC).In Wikipedia: The Free Encyclopedia. Retrieved from https://en.wikipedia.org/wiki/Stereographic_projection





© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

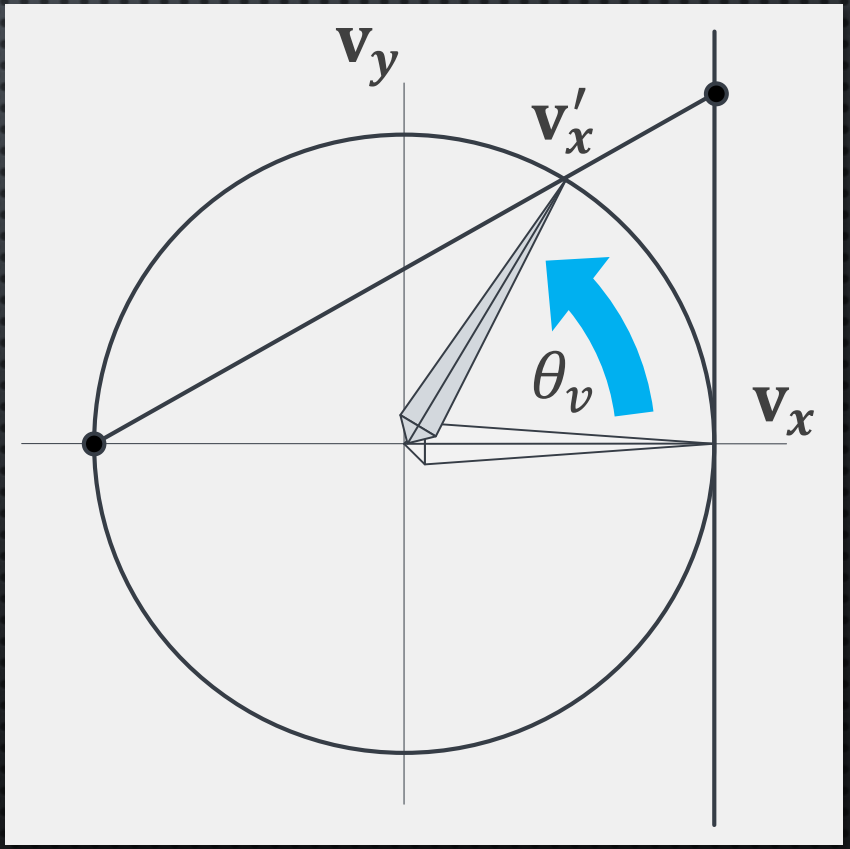
ステレオ投影による曲げ角度の分解・合成

初期姿勢の、ボーン方向、縦方向、横方向のローカルベクトルをそれぞれ $\mathbf{v}_x$ $\mathbf{v}_y$ $\mathbf{v}_z$ とする。

ボーンを目標方向 $\mathbf{v}'_x$ に向ける回転を、横方向の角度 θ_h と縦方向の角度 θ_v に分解。

$$\theta_h = -2\tan^{-1}\left(\frac{\mathbf{v}_z \cdot \mathbf{v}'_x}{\mathbf{v}_x \cdot \mathbf{v}'_x + 1}\right)$$

$$\theta_v = \quad 2\tan^{-1}\left(\frac{\mathbf{v}_y \cdot \mathbf{v}'_x}{\mathbf{v}_x \cdot \mathbf{v}'_x + 1}\right)$$



分解された角度 θ_h と θ_v から、方向ベクトル $\mathbf{v}'_x$ を合成。

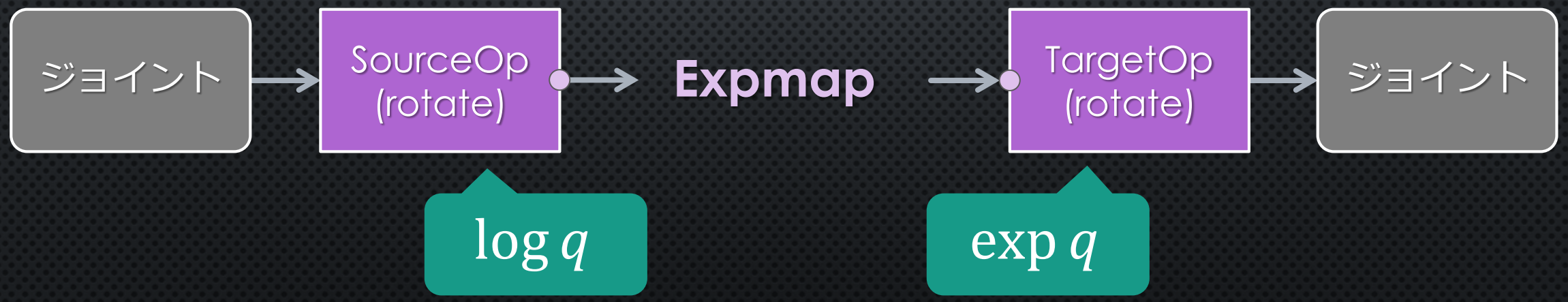
$$\mathbf{v}'_x = (s-1)\mathbf{v}_x + s\tan\left(\frac{\theta_v}{2}\right)\mathbf{v}_y + s\tan\left(-\frac{\theta_h}{2}\right)\mathbf{v}_z$$

where

$$s = \frac{2}{\tan^2\left(-\frac{\theta_h}{2}\right) + \tan^2\left(\frac{\theta_v}{2}\right) + 1}$$

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

分解と合成（扱いやすい値にする写像）



© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

Bend & Roll と Expmap の使い分け

Bend & Roll

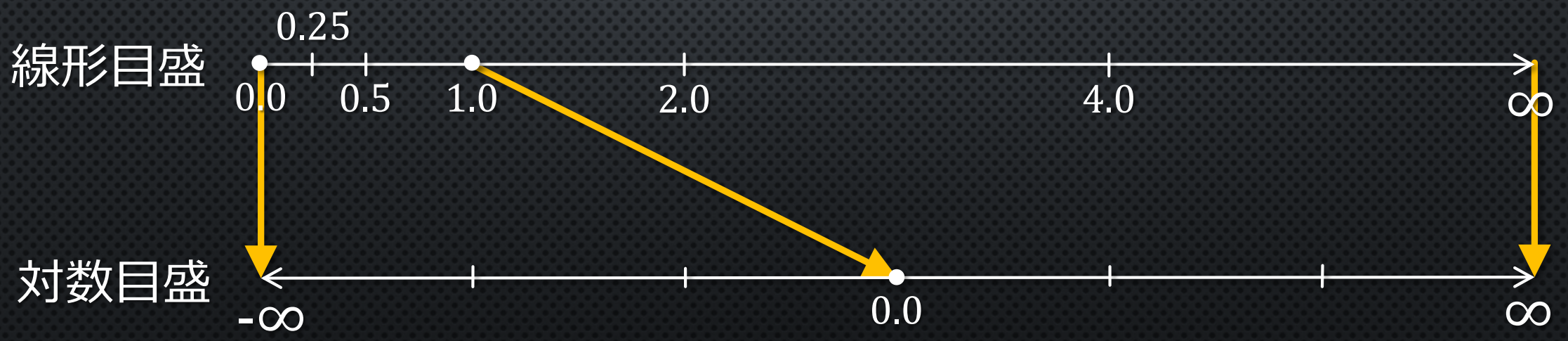
- 「曲げ」と「ひねり」には順番があり、部位ごとに適するものを使い分ける。
- 直感的に理解しやすく、ボーン回転の分析に適している。

Exponential Map

- 「曲げ」と「ひねり」のような順番がない。
- ボーン方向という概念に依存しないため、機械的に算出したものの制御方法として都合が良い。

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

扱いやすいスケール表現

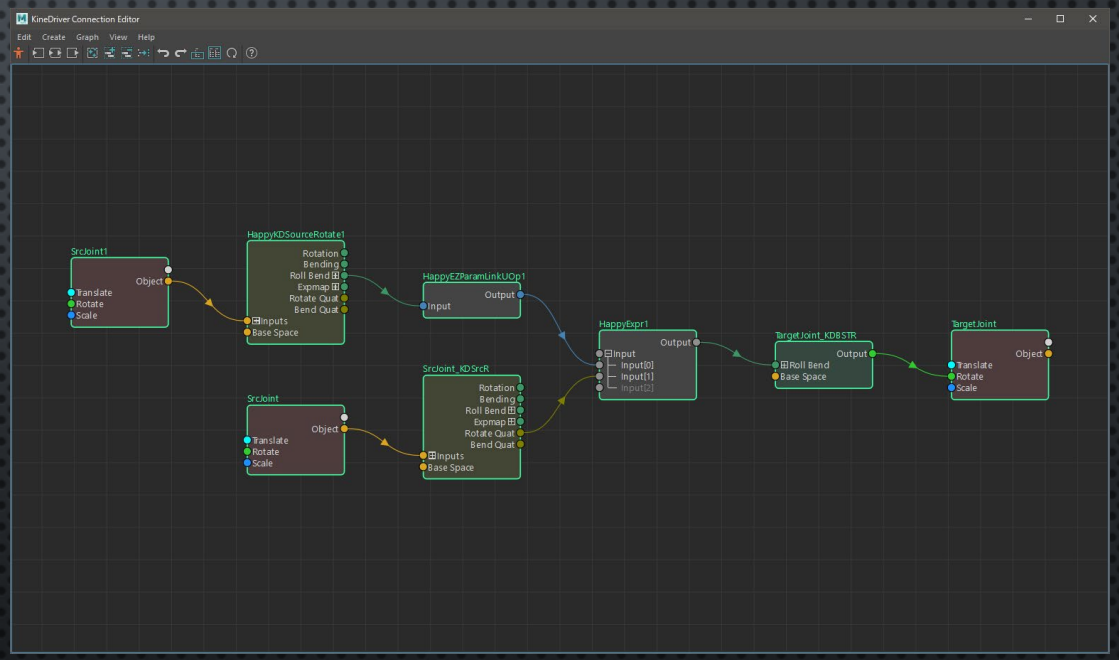


© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

エフェクターノードグラフ

Source 入力、Target 出力、及びエクスターナル入出力のリレーションは、以下の「エフェクター」ノードを自由に組み合わせて構築する。

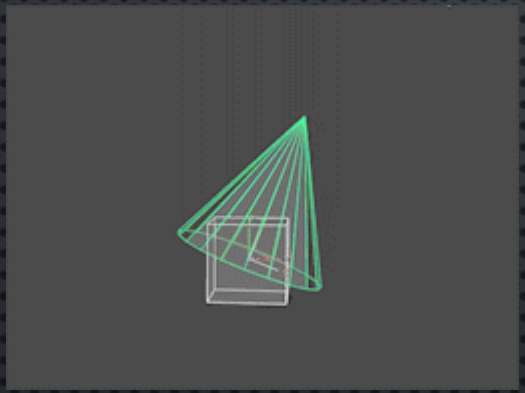
- EZParamLinkLinear（線形補間）
- EZParamLink（簡易ベジェ補間）
- LinkWith（ドリブンキー）
- RBFInterp（多次元ドリブンキー）
- Expr（任意の数式）



© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

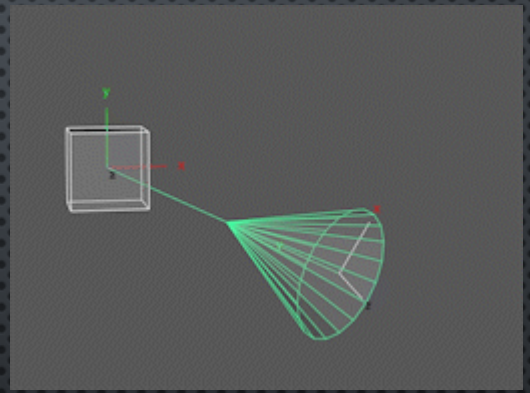
Constraint オペレータ

エフェクターノードグラフとは別分類の機能として、シンプルなコンストレイン機能を持つ。



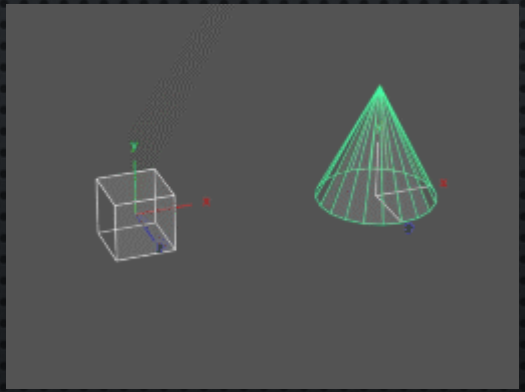
Position

位置の拘束。
１～ｎ個のソースをブレンド可能。



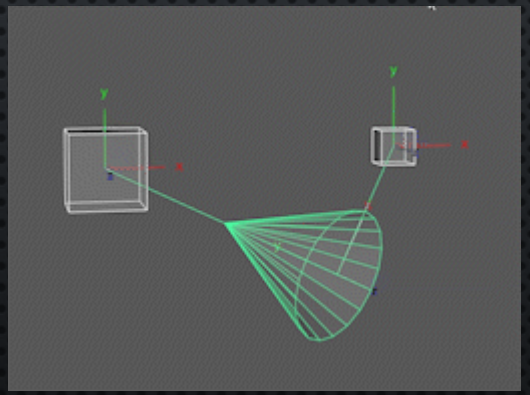
Direction (2点)

ローカル最短回転で対象方向を向く拘束。



Orientation

回転姿勢の拘束。
１～ｎ個のソースをブレンド可能。

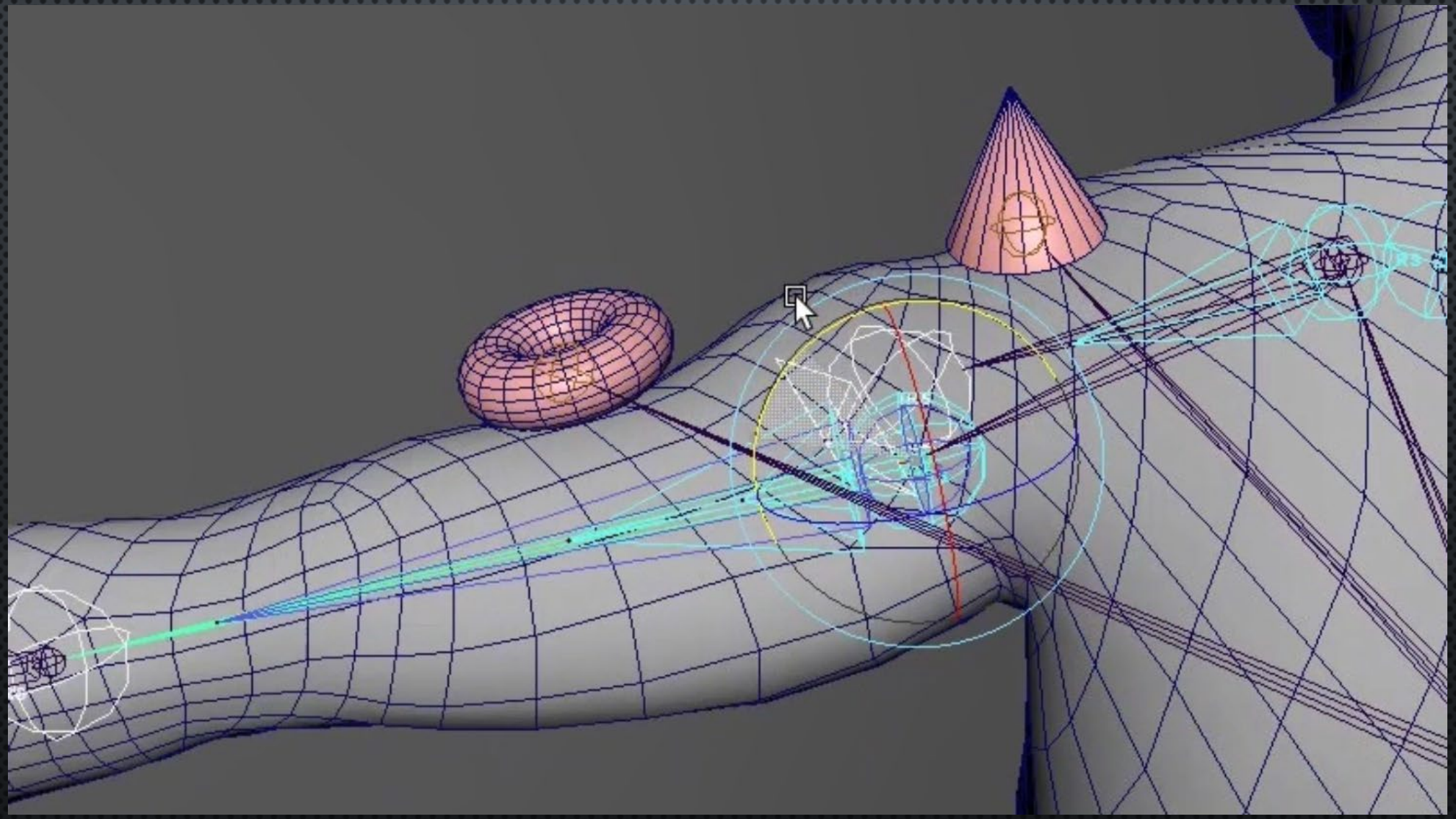


Direction (3点)

アップ方向も追加しひねり回転も制御。

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

メッシュアタッチ機能



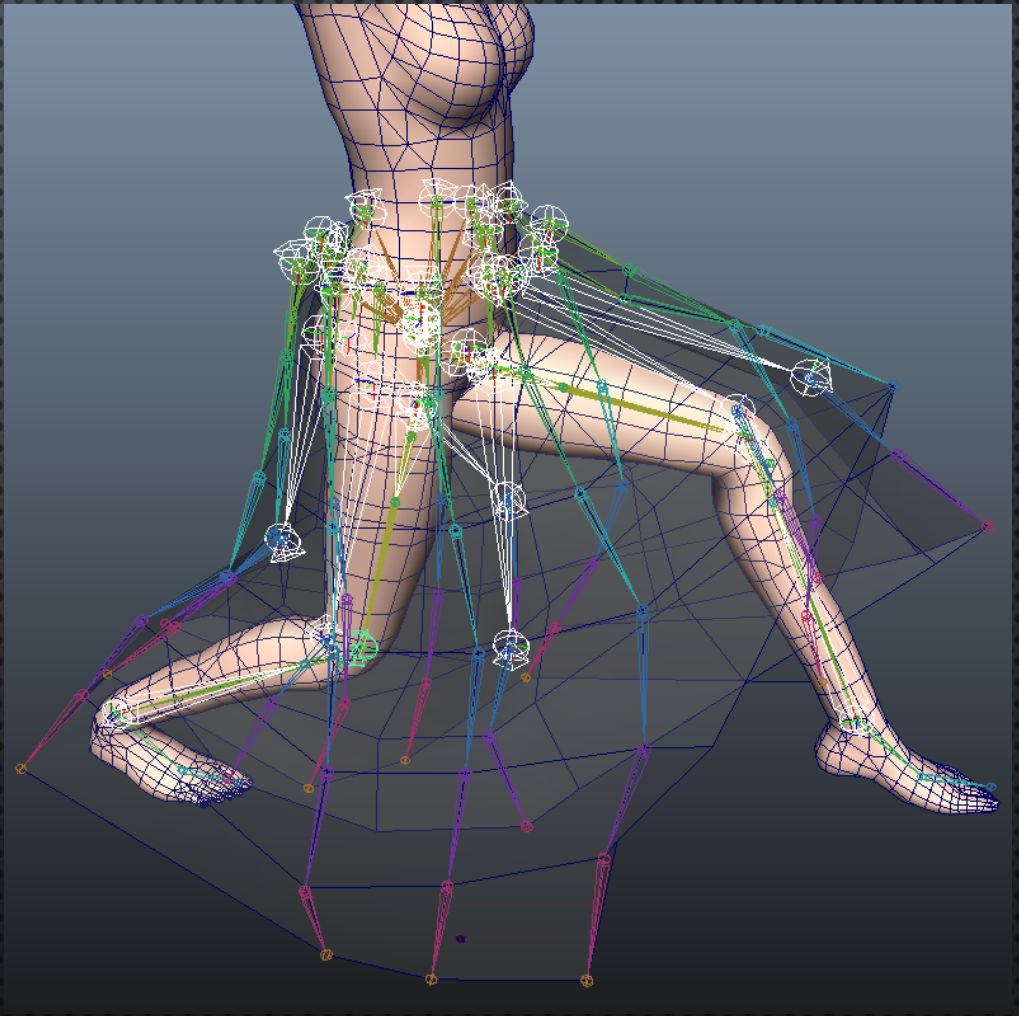
© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

メッシュアタッチ機能

補助骨をメッシュ表面に拘束する機能。

Position と Orientation コンストレインのブレンドウェイトを、アタッチする箇所のスキンウェイトから算出する。

Bonamik（揺れ骨システム）のガイド骨を作るのに、よく利用される。



© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

Maya® でのオーサリング

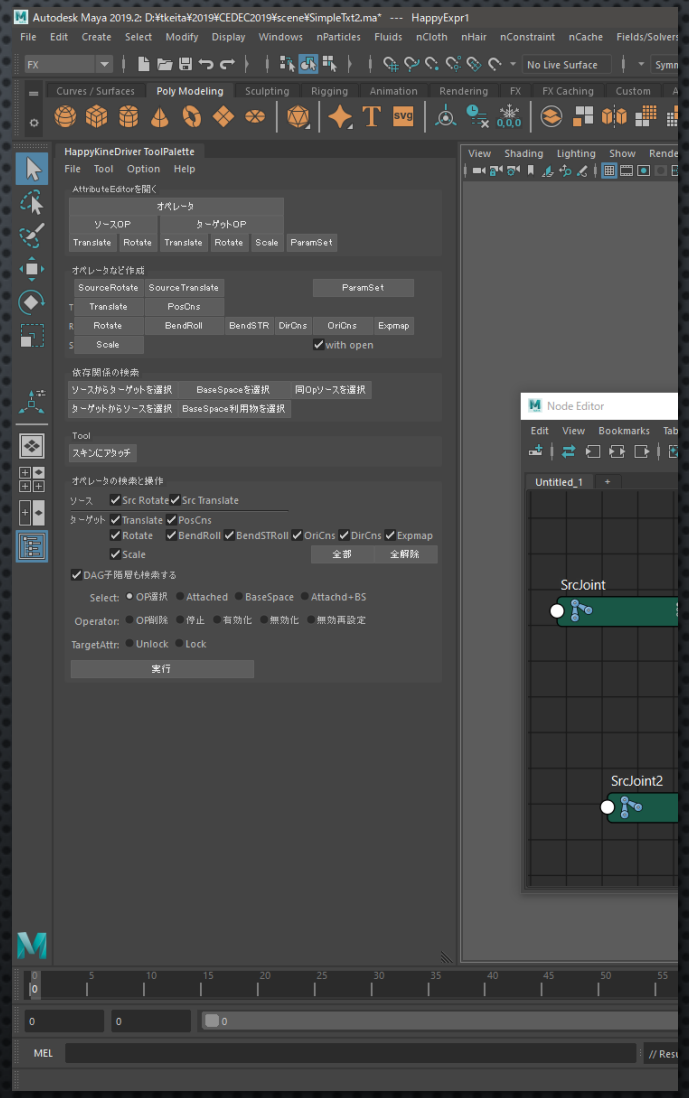
© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

各種エフェクターノードでの接続デモ

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

Maya® KineDriverプラグイン構成

- ノードプラグイン （C++実装）
  - SourceOp, TargetOp, Effector
    シンプルな 計算のみの DependencyNode
- Pythonスクリプト
  - KDI ファイルインポーター、エクスポーター
  - ツールパレット GUI
  - アトリビュートエディタ GUI
  - 補助ツール
    （スキンアタッチ機能、ミラー機能等）



© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

Maya®上での単位

SourceOp, TargetOp:

角度と距離は単位をもつ Angle 型、Distance 型のアトリビュート

Effector:

単位のない Unitless 型のアトリビュート

- 接続時に UnitConversion ノードが挿入される
- Effector は使い慣れた単位のまま編集
- 計算は内部単位（Radian, Centimeter）

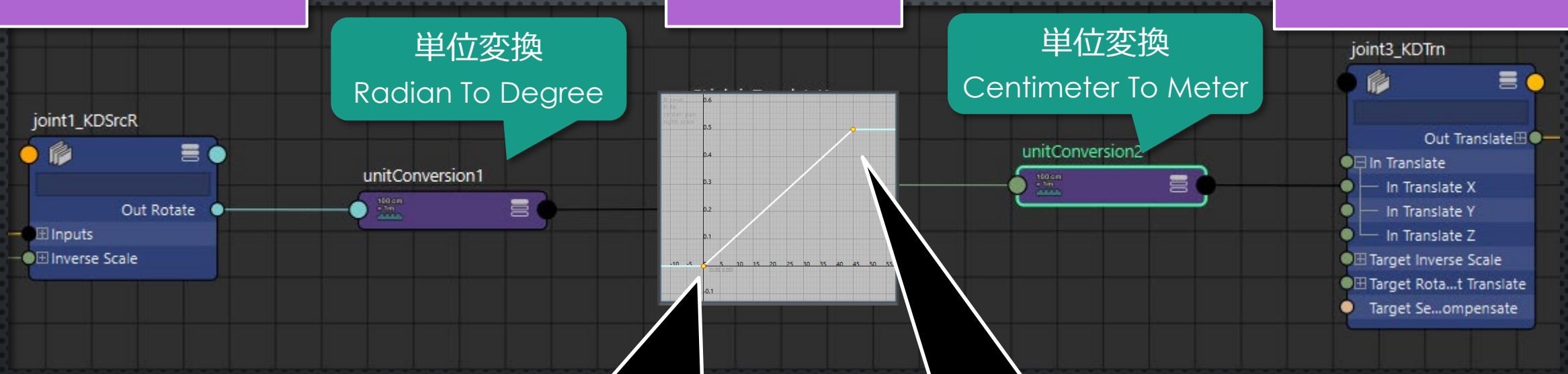
© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

単位の取り扱い

SourceOp
角度のパラメータ

Effector
単位なし

TargetOp
距離のパラメータ



UI表示: 45° (Degree)
内部値: 0.785 (Radian)

In: 0 , Out: 0

In: 45, Out: 0.5

UI表示: 0.5m
内部値: 50cm

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

中間ファイルについて

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

KDI ファイルフォーマット

- XML形式
- SourceOp, TargetOp, Effectorの各ノード情報
- パラメータ同士の Connection 情報
- 評価順にデータ出力
  - ランタイムはこの順番に評価


```xml
<?xml version="1.0" ?>
<HappyKineDriverInfo version="3.7.0">
    <header>
        <user>tkeita</user>
        <scaleType>STANDARD</scaleType>
        <tool>Maya</tool>
    </header>
    <nodes>
        <source name="joint2.KDSrcR1" type="HappyKineDriverSourceRotate">
            <params>
                <sourceObjects count="1">
                    <sourceObject>
                        <object>joint2</object>
                        <weight>1.0</weight>
                    </sourceObject>
                </sourceObjects>
                <neutralRotate>
                    <x>0</x>
                    <y>0</y>
                    <z>0</z>
                </neutralRotate>
                <aimVector>
                    <x>1</x>
                    <y>0</y>
                    <z>0</z>
                </aimVector>
                <upVector>
                    <x>0</x>
                    <y>1</y>
                    <z>0</z>
                </upVector>
                <segmentScaleCompensate>1</segmentScaleCompensate>
                <reverseOrder>0</reverseOrder>
            </params>
        </source>
        <effector name="KDRBF1" type="RBFInterp">
            <params>
                <input>
                    <value index="0">0.412968260011</value>
```


© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

KDI (中間ファイル)入出力時のスケーリング

- KDIインポート・エクスポートの時にスケールを指定可能
- Distance 型の固定パラメータ
- SourceOp の Distance 型からのコネクション
- TargetOp の Distance 型へのコネクション

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

Unreal® Engine 4 プラグイン

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

開発の流れ

- Unreal® Engine 4 採用ゲームプロジェクトのプログラマが元実装
- ゲームプロジェクト依存の部分の切り分け
- プラグイン化
- その後、要望に応じた変更、修正、機能追加など

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

モジュール構成

- KineDriverEd （エディタモジュール）
  - KDI File Import
  - Preview Support
  - AnimGraphNode
- KineDriverRt （ランタイムモジュール）
  - KDI Asset Data
  - Runtime Data & Compute Code
  - Component
  - AnimNotify
  - AnimNode

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

Plug-in 化について

- DCC ツールのプラグインを念頭
- 再利用性を高めることで他のプロジェクトで導入しやすく
- バイナリプラグインは社内では意義は低かった
  - 基本的にソースからビルドされるため

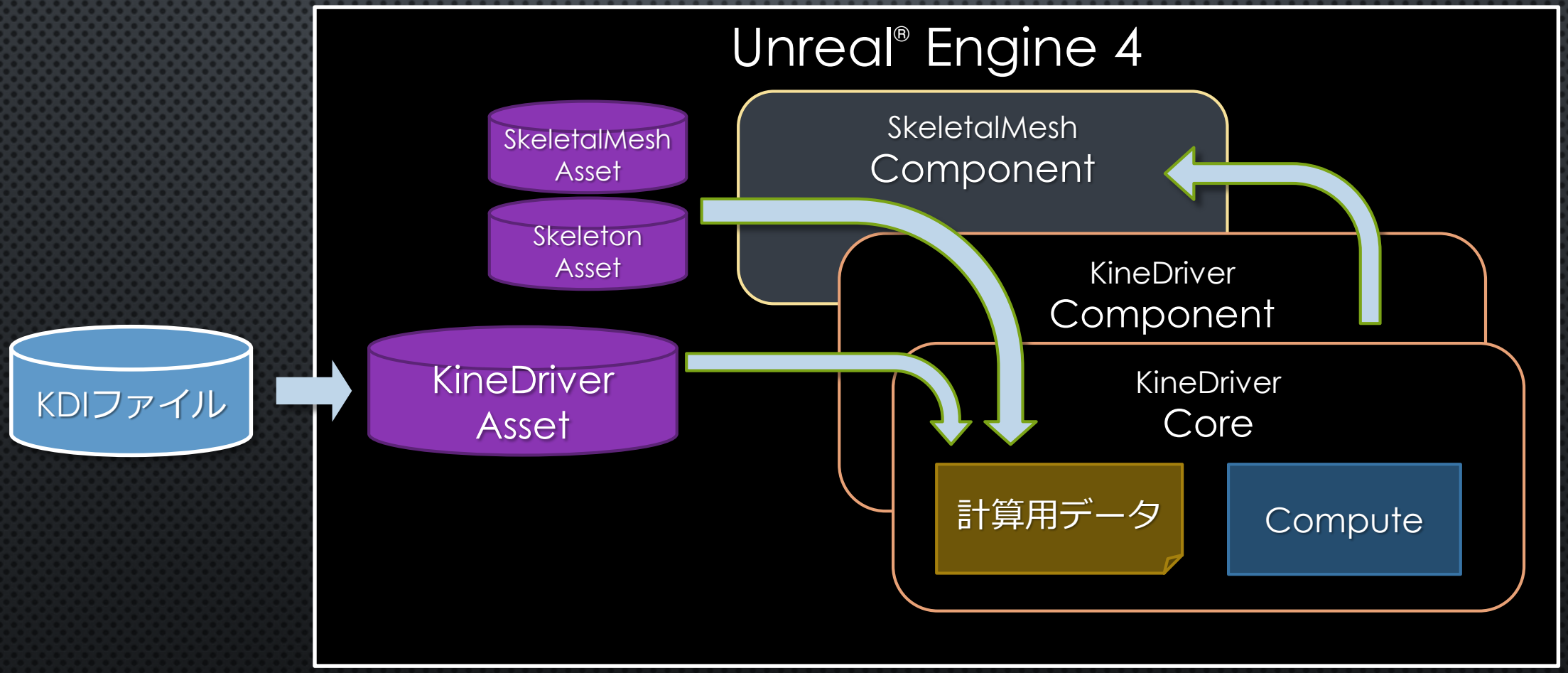
© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

Component と AnimNode

- AnimNode 版
  - 標準機能の SkeletalControl を踏襲
  - AnimationEditor で KineDriver をプレビューできない
- Component 版
  - SkeletalMesh コンポーネントの子に配置
  - Tick で SkeletalMesh コンポーネントの Transform を書き換える
  - AnimationEditor 上で KineDriver をプレビューできる

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

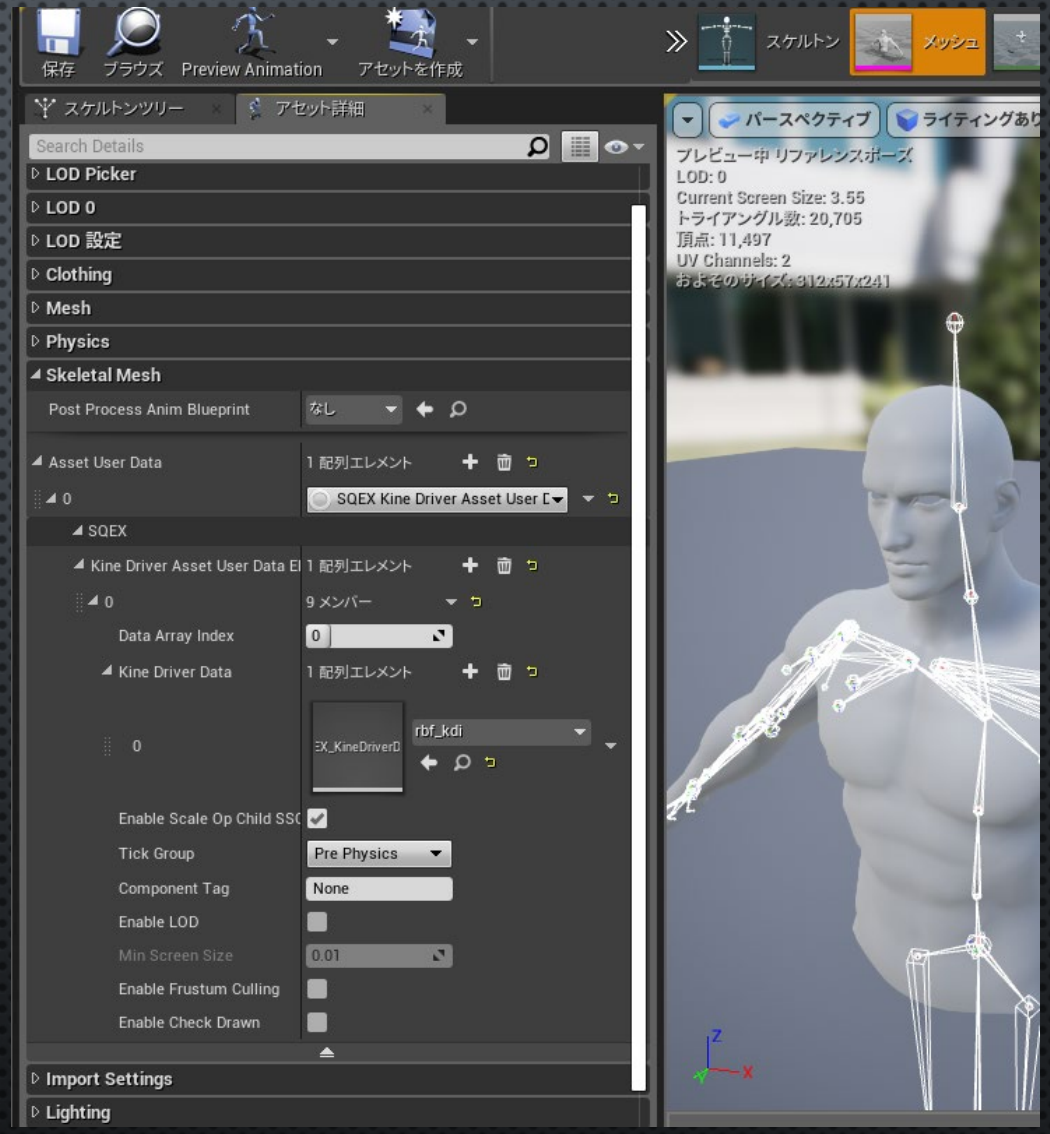
処理の流れ



© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

AssetUserData

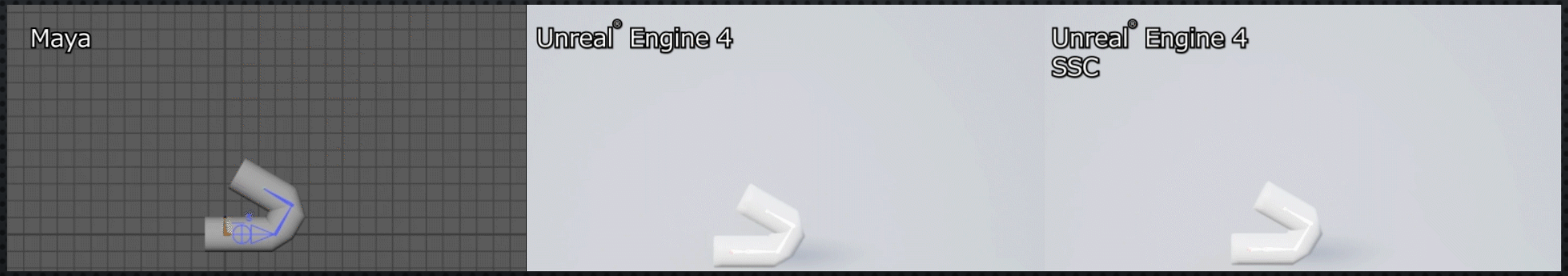
- SkeletalMesh アセットに任意の情報を持たせる仕組み
- KineDriver がどの KDI Asset を使うかなどの設定や、各種オプション
- 開発の運用ルール上、デザイナーが Component を変更せずに、SkeletalMesh の変更だけで済むようにする目的



© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

Scale

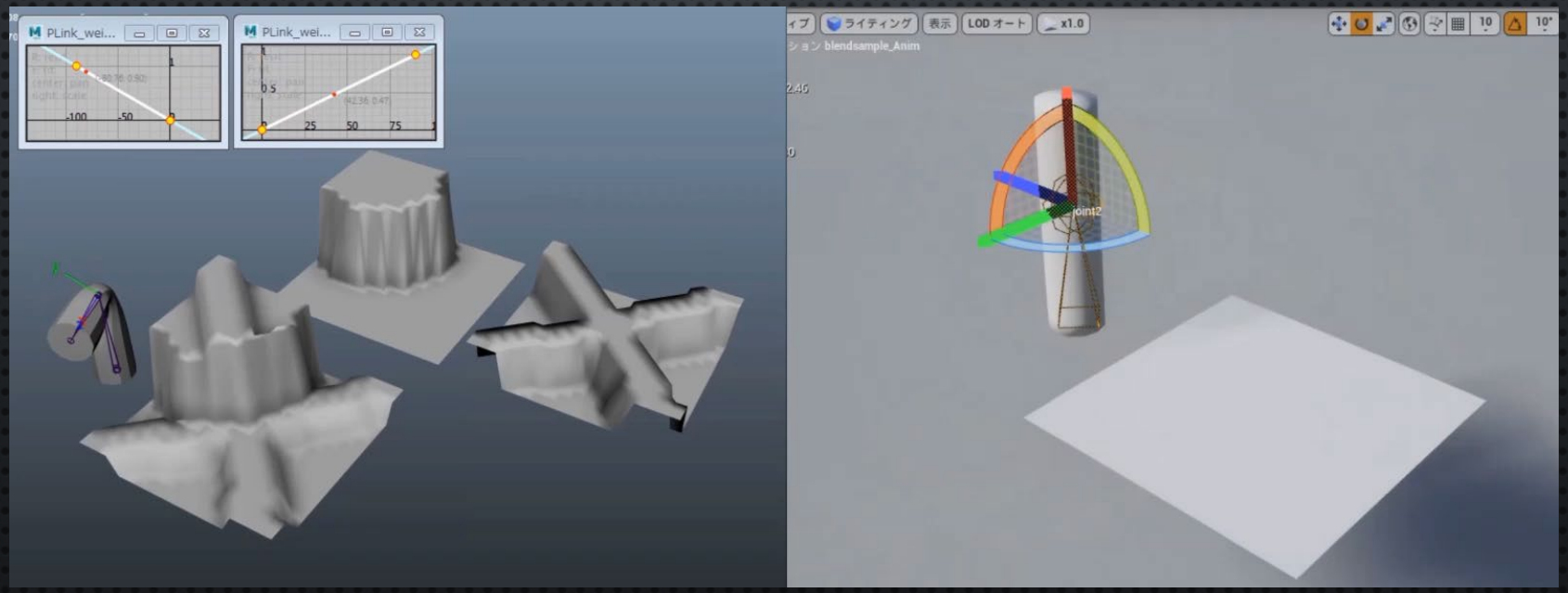
- Maya®はセグメントスケール補正 → 子に伝搬しない
- Unreal® Engine 4 は階層スケーリング → 子に伝搬する
- KineDriver TargetScaleOp で Scale補正
  - Unreal® Engine 4 版では、TargetScaleOp の対象 Bone の子で、セグメントスケール補正を行う
  - Maya®と同じ結果が得られる



© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

BlendShape

- Maya® BlendShape
  - SourceOp から BlendShape のウェイトアトリビュートへ接続
- Unreal® Engine 4
  - KineDriver から TargetBlend へ値を流し込む



© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

AnimNotify

- AnimationEditor 上で AnimNotify を配置
- KineDriver を一時的にON/OFF

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

Bonamikとの評価順制御など

- 別々のプラグイン 別々のコンポーネント
  - Unreal® Engine 4 の AddTickPrerequisiteComponentで
    コンポーネント同士の実行順を制御できる
  - Kine Bona が別々であるメリット
    - 開発担当者が異なる
    - それ以外のコンポーネントの開発も想定される
- 統合コンポーネント
  - 実際に KineDriver Bonamik を採用したゲームプロジェクトでは、
    プロジェクト側で KineDriver Bonamik の統合コンポーネントが作られた
  - マルチスレッドタスク数の削減による最適化
  - 統合した AssetUserData
    - デザイナーが実行順や複数適用など決定可能

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

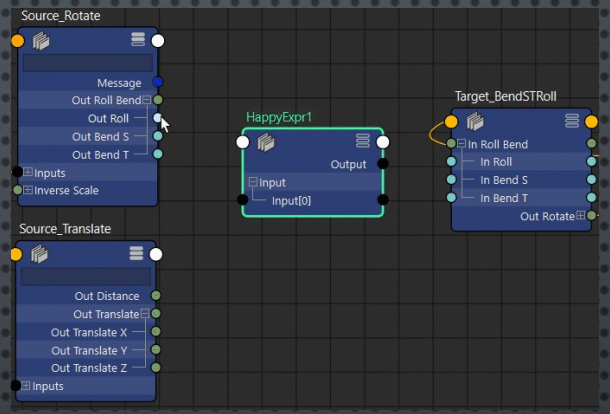
諸機能の実装

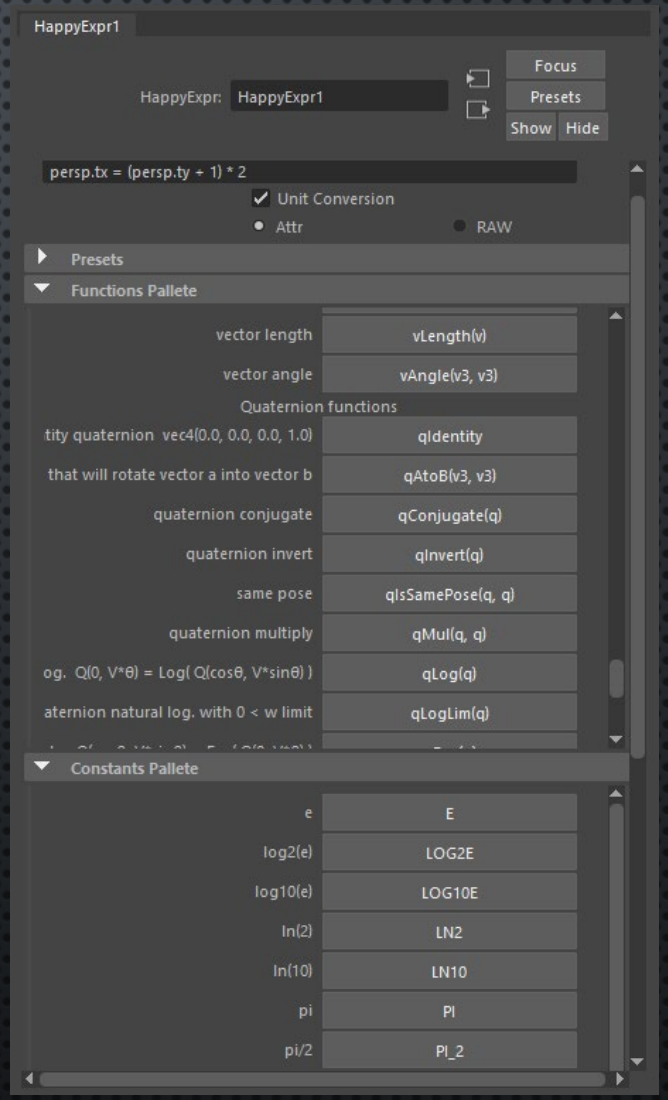
エクスプレッション

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

機能

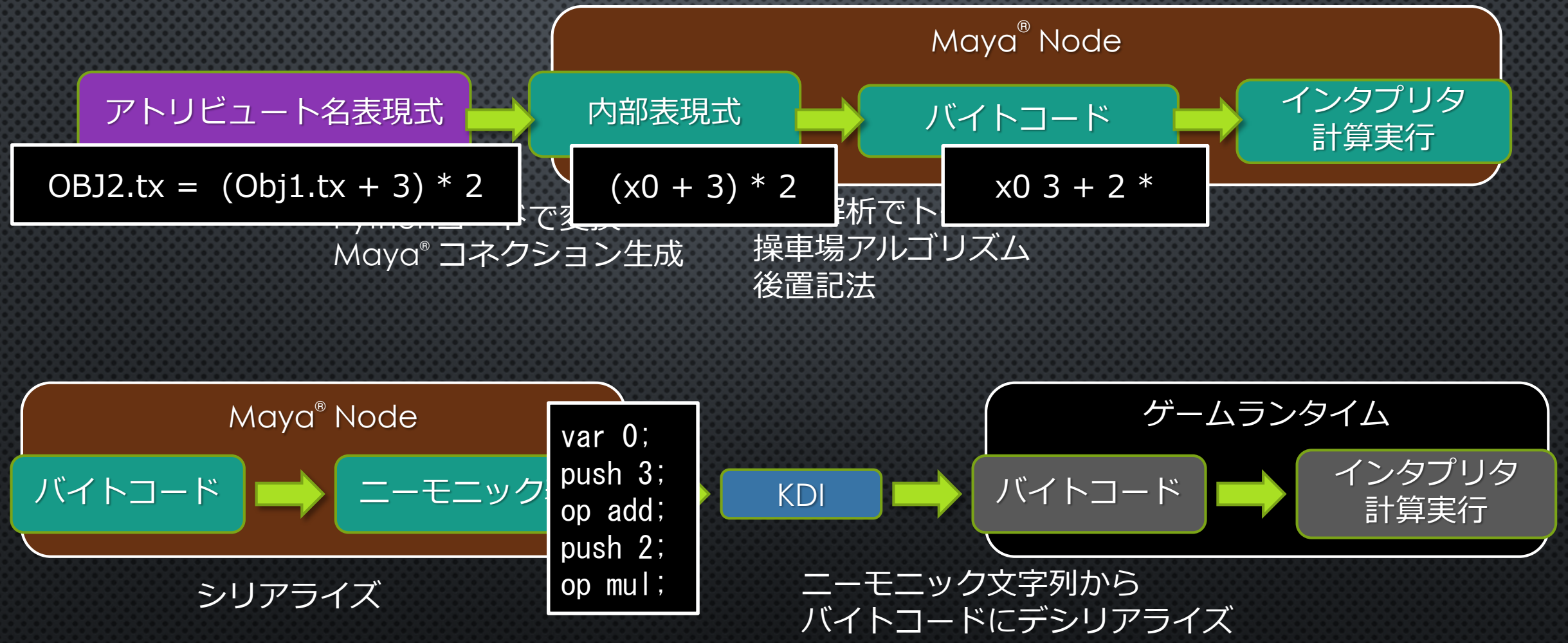
- 複数値の入力
  - 変数でアクセス
  - x0, x1, … xN
- シンプルな文法
  - 単一の式のみ
  - For など制御構文はナシ
- 四則演算とカッコ 「+ - * / ( ) 」
- ビルトイン関数 「sin, log, max, … 」
- ３次元ベクトルやクオータニオンも使える





© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

処理の流れ



© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

開発の経緯

- 条件分岐や合成をしたいという要望が多かった
  - ２つのコネクションで大きい方…
  - ２つのアングルの合計…
- KineDriver V3 では複雑なノード接続を可能とした
  - V2 では Source と Target のパラメータ間をつなぐのみ
- 初期想定では多種多様な演算ノードを用意し
  自由な計算を実現することを検討したが・・・
- GUI でノード配置や接続をするのは面倒
- 式で書きたい
- たくさんのノードを作るより自由度が高くシンプル

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

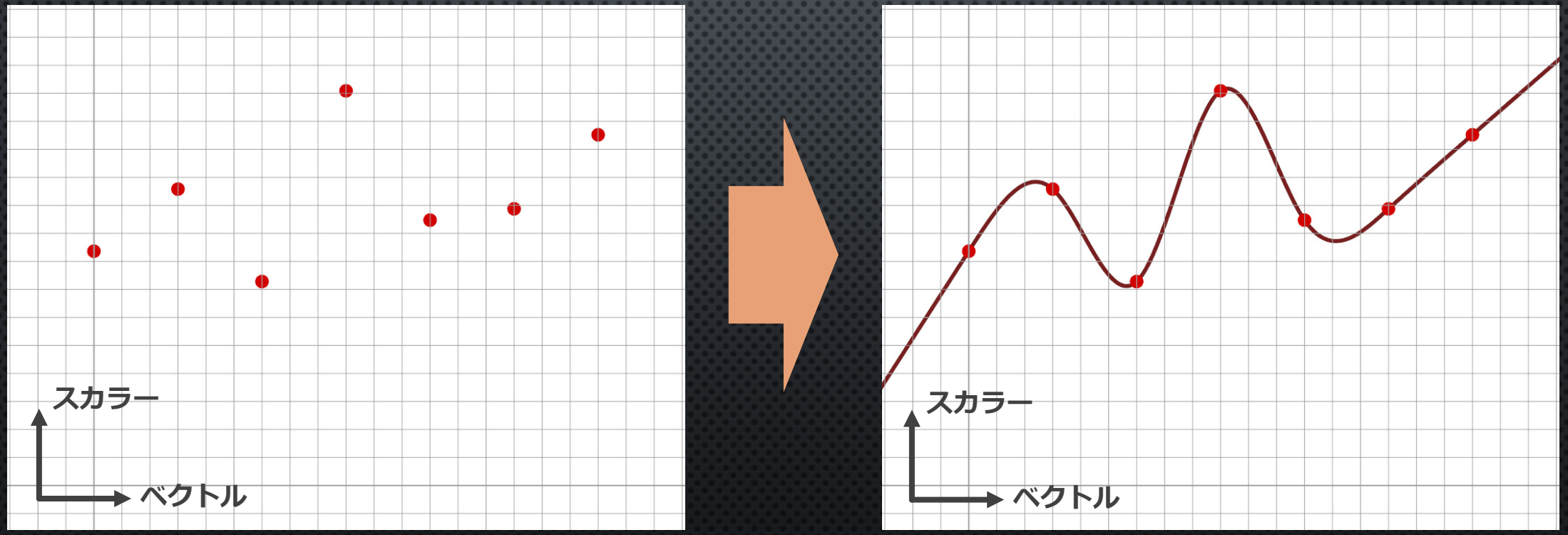
諸機能の実装

RBF 補間

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

RBF 補間とは

N 個の点列 $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ を通る滑らかな関数を得るフィッティング手法。
$\mathbf{x}_i$ は d 次元のベクトルで y_i はスカラー。



© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

RBF 補間の考え方

Radial Basis Function（放射基底関数）とは、中心 $\mathbf{c}$ からの距離のみに依存する関数 ϕ 。距離なので何次元ベクトルの入力にも使える。

$$\phi(\|\mathbf{x}-\mathbf{c}\|)$$

RBF 補間関数 $f(\mathbf{x})$ は N 個のキーの点列 $\mathbf{c}_i$ による ϕ の線形結合。

$$f(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi(\|\mathbf{x}-\mathbf{c}_i\|)$$

つまり、いい感じの重み $\mathbf{w} = [w_1 \quad \cdots \quad w_N]^{\mathrm{T}}$ を決定してあげれば良い。

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

ウェイトの決定方法

補間関数 $f(\mathbf{x})$ が全てのキー点を通るという条件を与える。

関数が j 番目のキー $(\mathbf{x}_j, y_j)$ を通るために重み $\mathbf{w}$ が満たすべき条件式を考える。

$$y_j = \sum_{i=1}^{N} w_i \phi(\|\mathbf{x}_j - \mathbf{c}_i\|)$$

これを $j = 1, \ldots, N$ の全てのキーに対して考えると $N \times N$ の行列表記にできる。

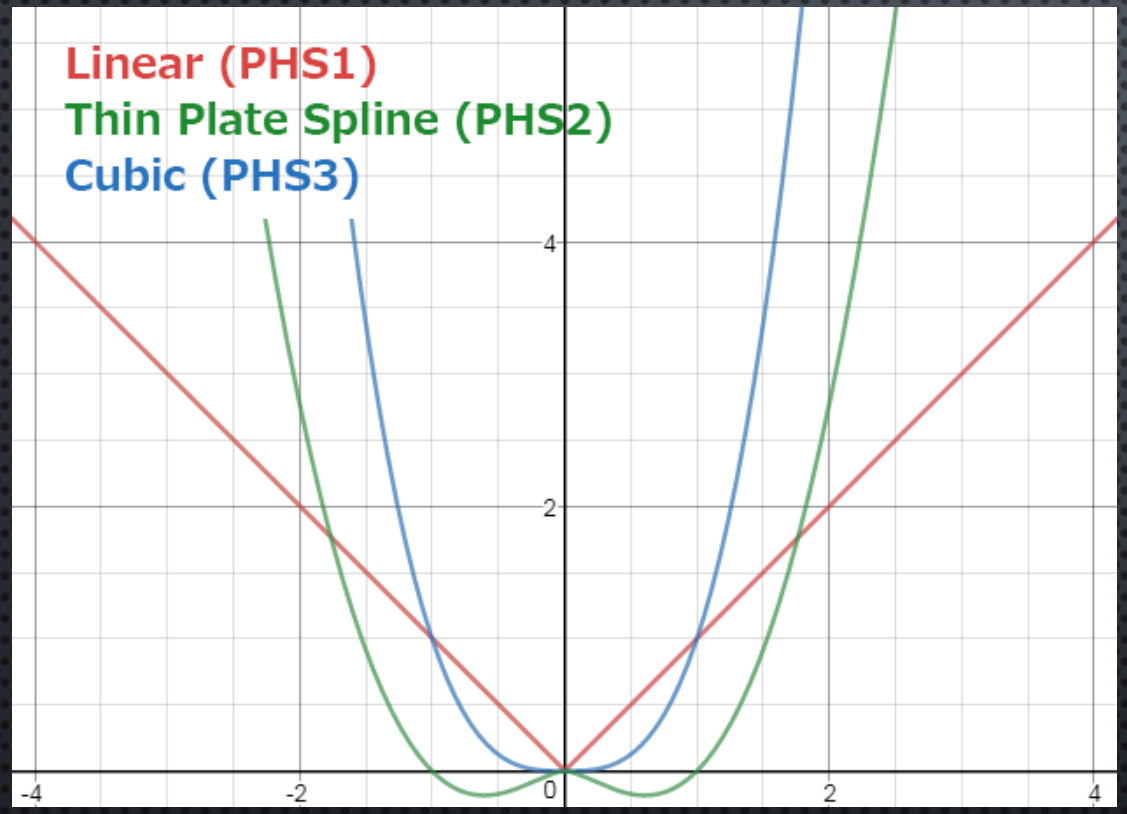
$$\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} \phi(\|\mathbf{x}_1 - \mathbf{c}_1\|) & \ldots & \phi(\|\mathbf{x}_1 - \mathbf{c}_N\|) \\ \vdots & \ddots & \vdots \\ \phi(\|\mathbf{x}_N - \mathbf{c}_1\|) & \ldots & \phi(\|\mathbf{x}_N - \mathbf{c}_N\|) \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix}$$

この連立一次方程式を解くことで重み w_i が求まる。

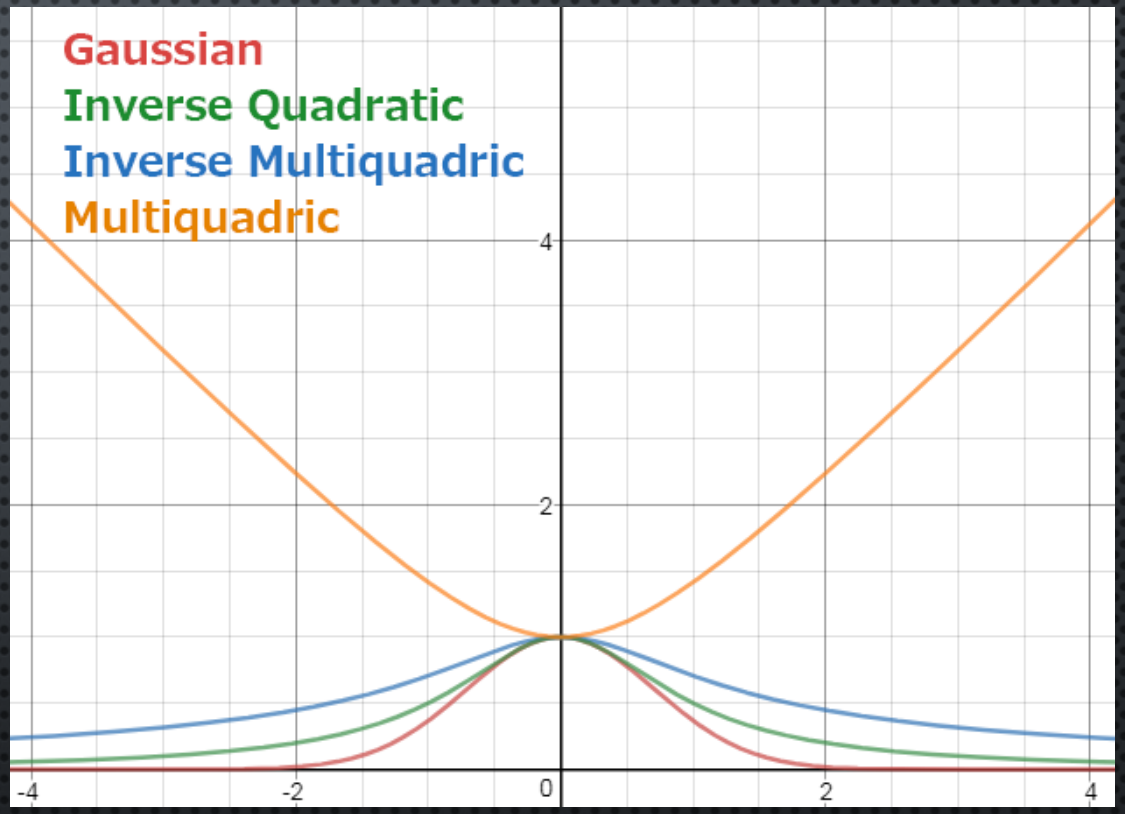
ランタイムでの補間計算では、オーサリング時に求めた重みを使うので軽い。

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

RBF の種類



Polyharmonic Spline
（重調和スプライン）



Polyharmonic Spline 以外
（上向きの Multiquadric は他と性質が異なる）

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

RBF の数式の比較

中心 $\mathbf{c}$ からの距離を $r = \|\mathbf{x} - \mathbf{c}\|$ とする。

Polyharmonic Spline :

$$\phi(r) = \begin{cases} r^k & with\ k = 1,3,5,\ldots \\ \mathrm{r}^{\mathrm{k}}\ln(r) & with\ k = 2,4,6,\ldots \end{cases}$$

チューニングのための ϵ の乗算が不要
（補間結果にほぼ影響しない）

Multiquadric :

$$\phi(r) = \sqrt{1 + (\epsilon r)^2}$$

Gaussian :

$$\phi(r) = e^{-(\epsilon r)^2}$$

Inverse Quadratic :

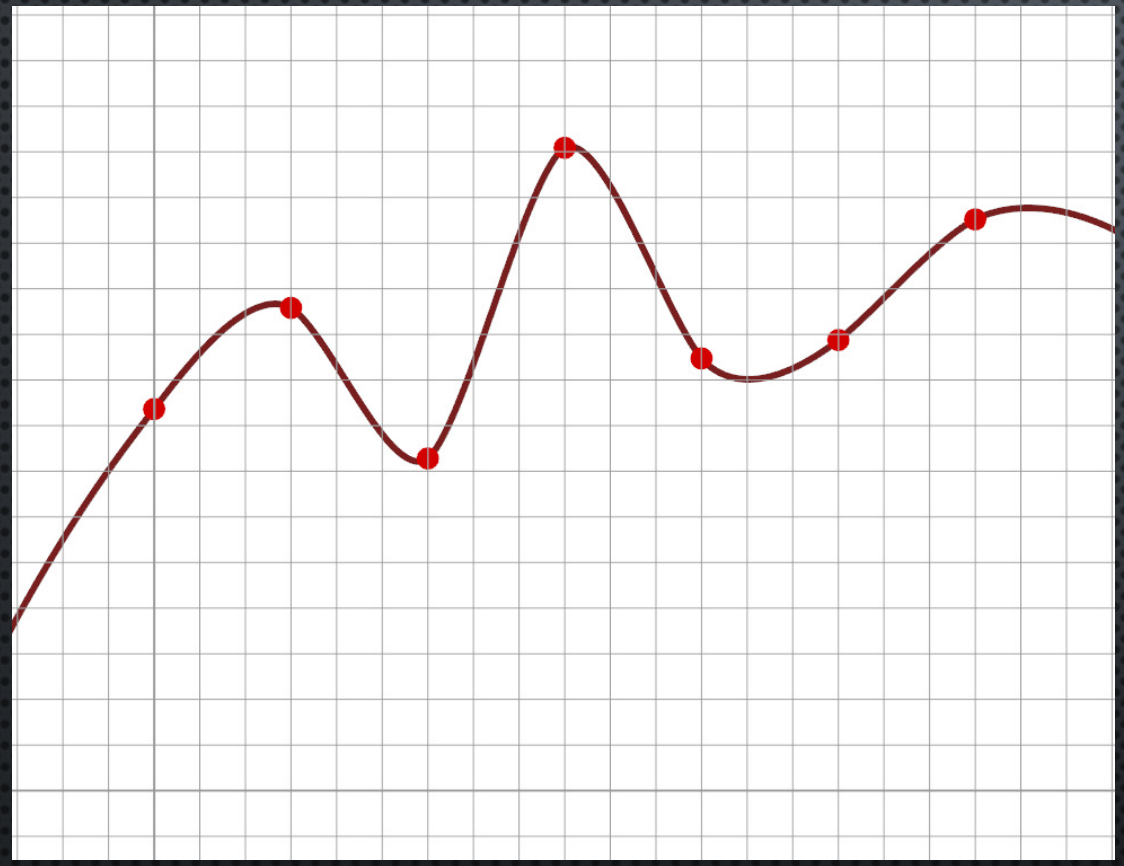
$$\phi(r) = \frac{1}{1 + (\epsilon r)^2}$$

Inverse Multiquadric :

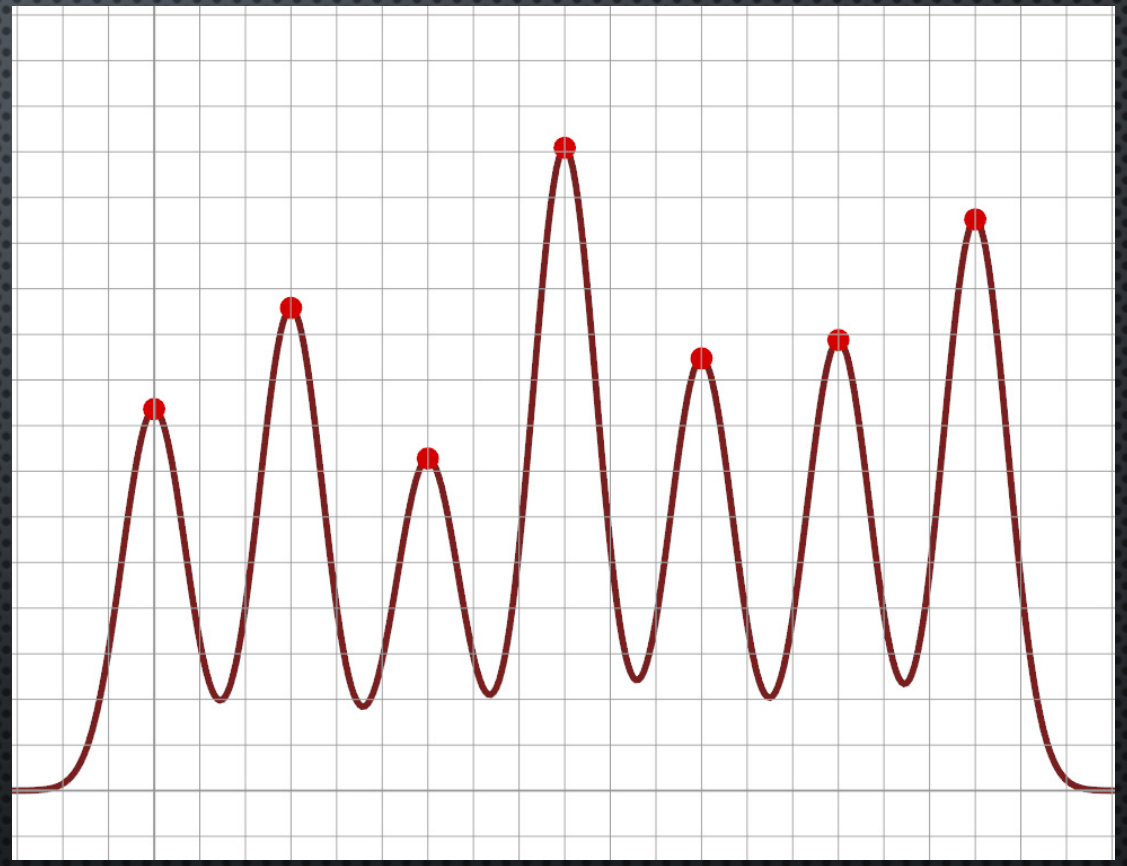
$$\phi(r) = \frac{1}{\sqrt{1 + (\epsilon r)^2}}$$

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

RBF 補間の結果比較



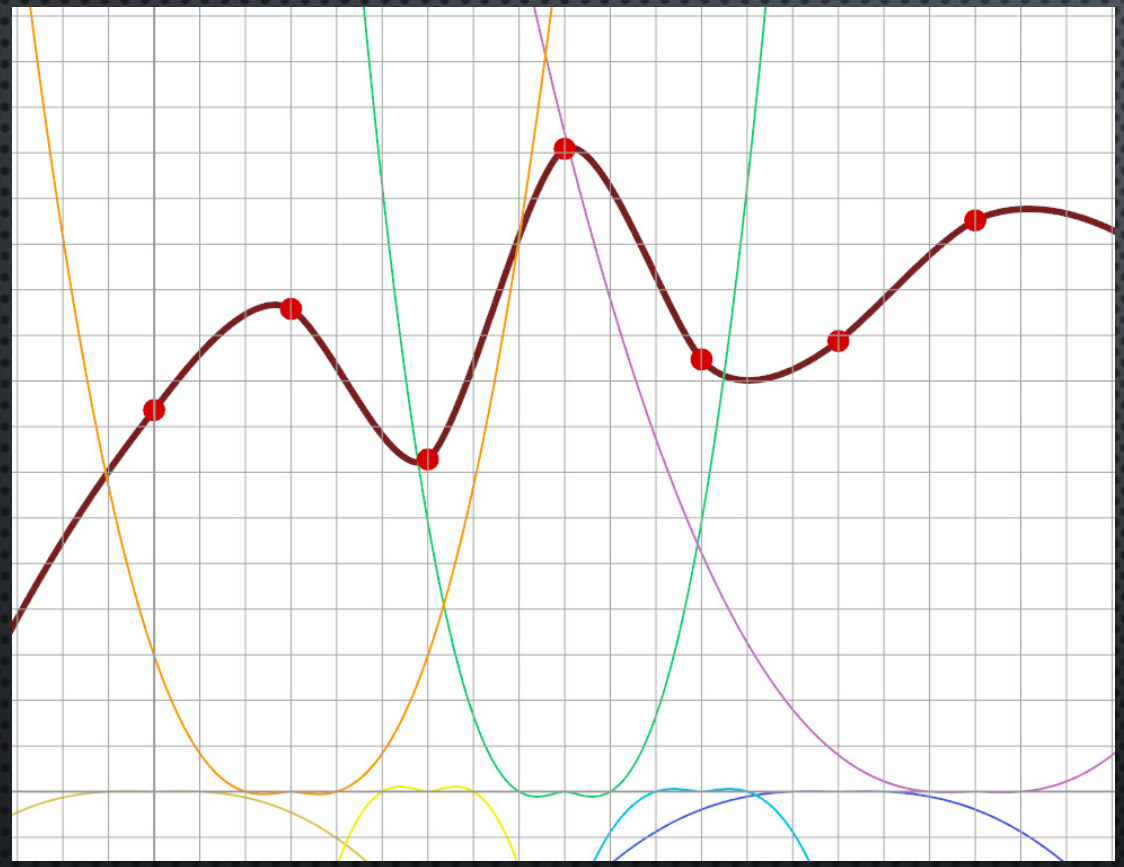
Thin Plate Spline
（二次の Polyharmonic Spline）



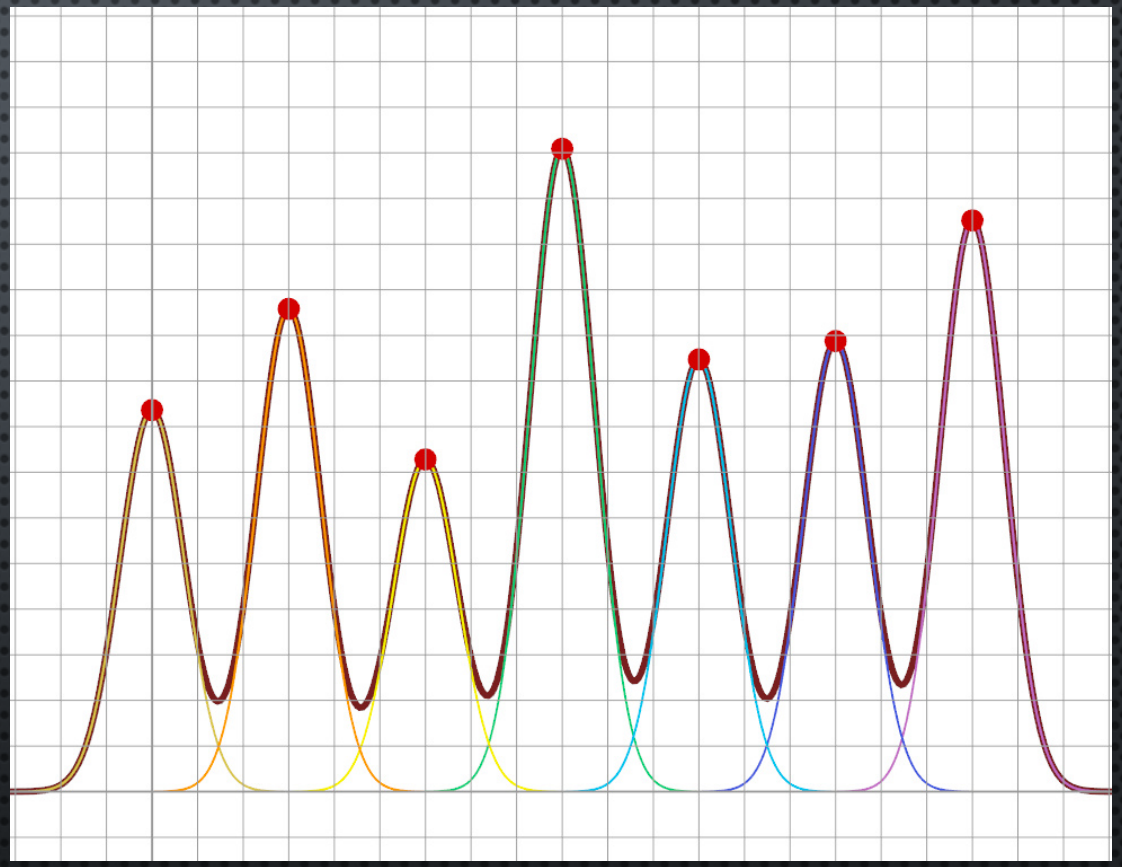
Gaussian

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

RBF 補間の結果比較（各キー RBF の内訳）



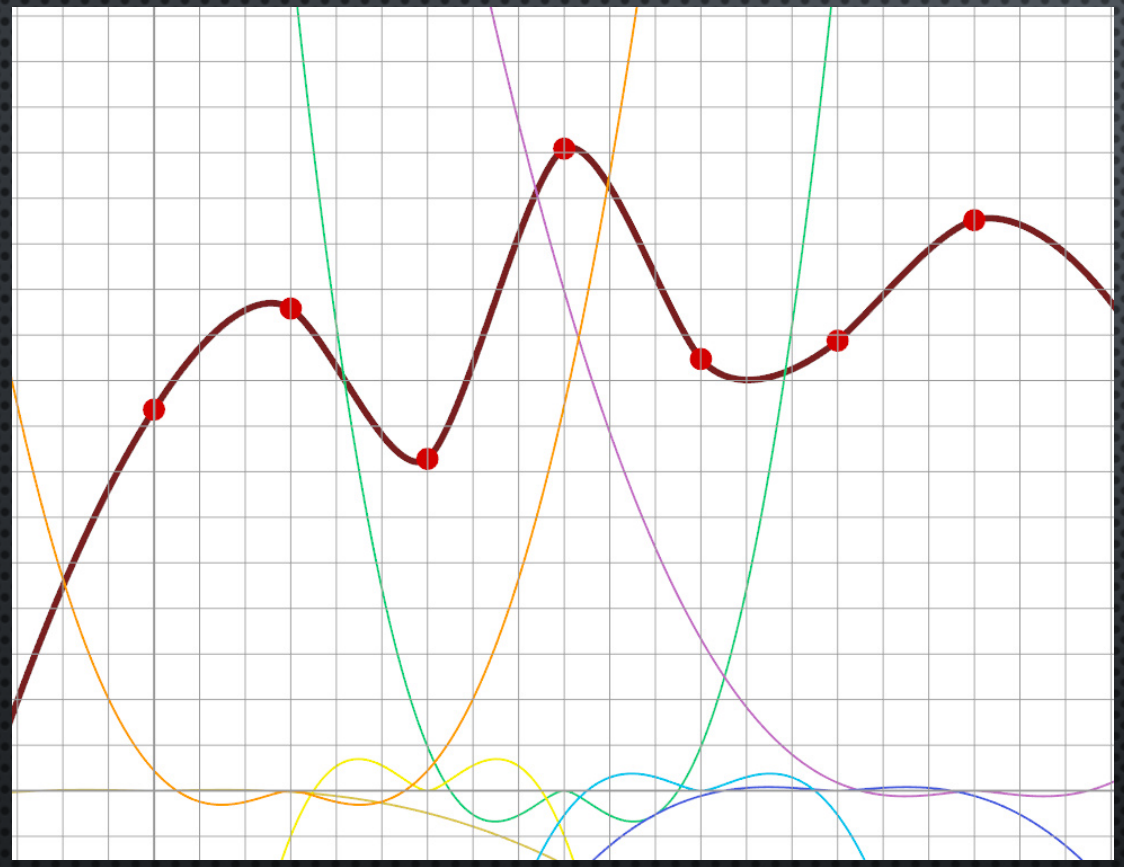
Thin Plate Spline
（二次の Polyharmonic Spline）



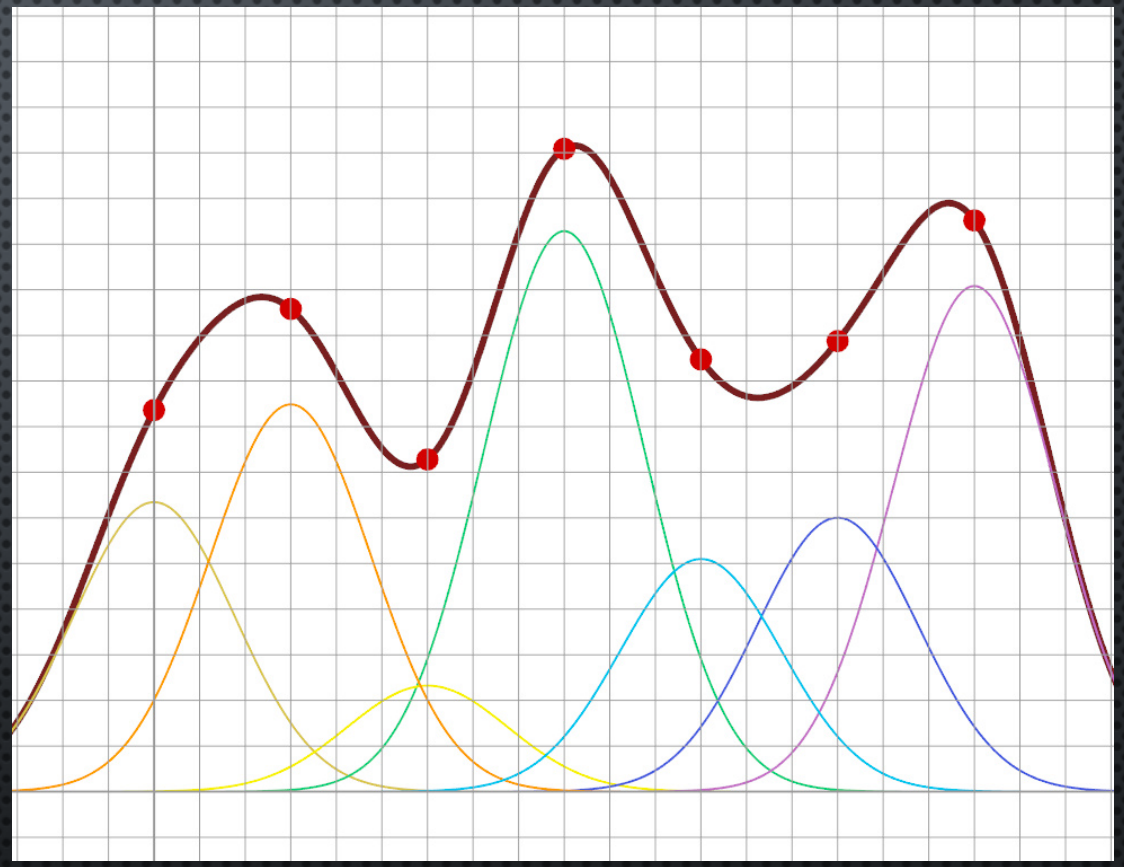
Gaussian

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

RBF 補間の結果比較（影響範囲の調整）



Thin Plate Spline
（二次の Polyharmonic Spline）



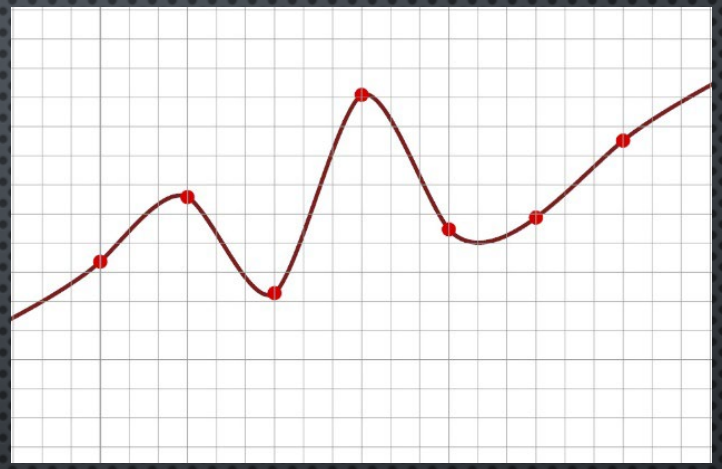
Gaussian

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

RBF の使い分け

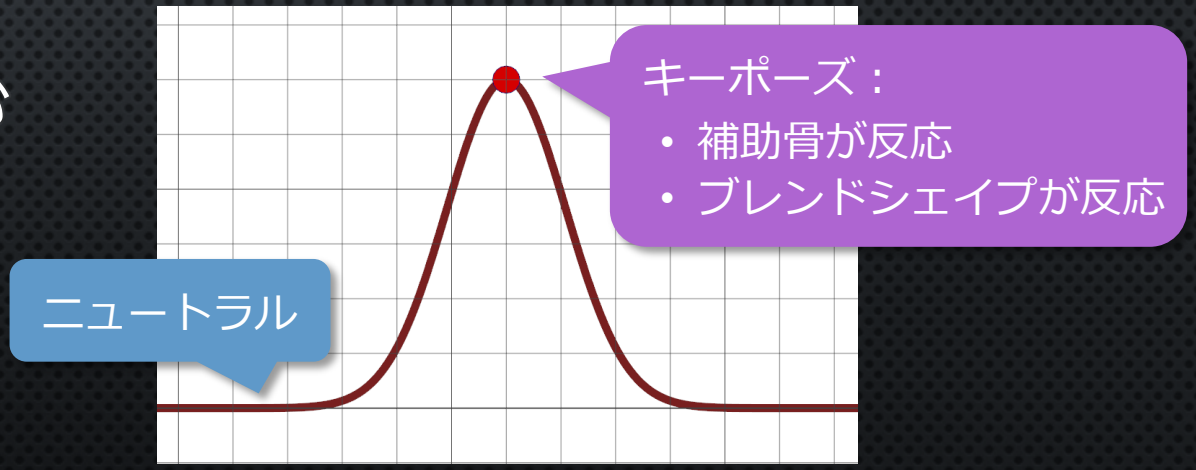
Polyharmonic Spline

調整なしで
いい感じに補間したい場合に。



Gaussian など

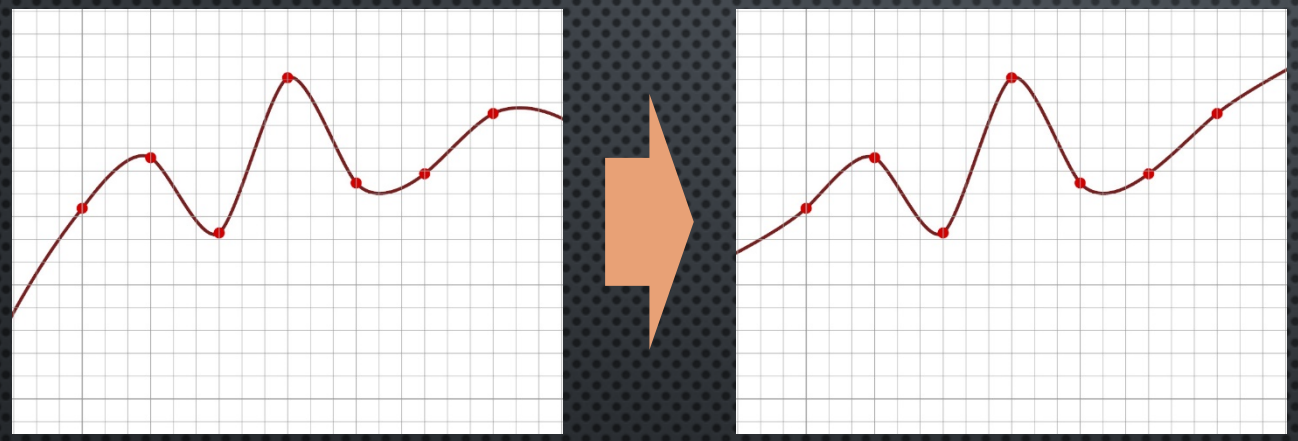
キー間がニュートラルに落ち込む
特性をあえて利用したい場合に。

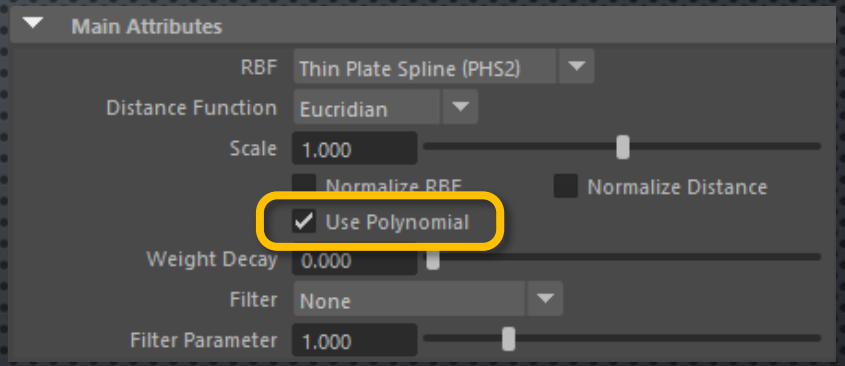


© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

線形補間の併用による補間の改善

RBF補間では、通常期待されるような外挿（Extrapolation）は得られないが、補間関数に 線形項を追加 することで内挿と外挿をやや改善する。





$$f(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi(\|\mathbf{x} - \mathbf{c}_i\|) + \mathbf{v}^{\mathrm{T}} \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}$$

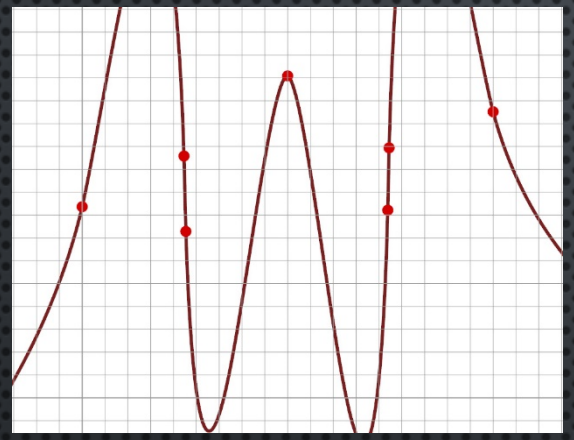
（ $\mathbf{v}$ は $\mathbf{x}$ の次元+1 個の追加のウェイト）

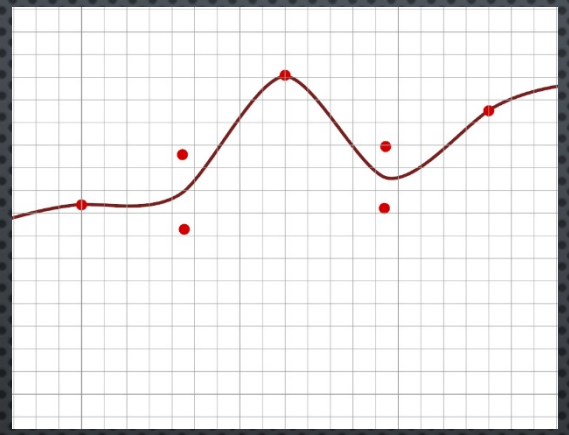
条件： $\sum_{i=1}^{N} w_i = 0\,,\qquad \sum_{i=1}^{N} w_i \mathbf{c}_i = \mathbf{0}$

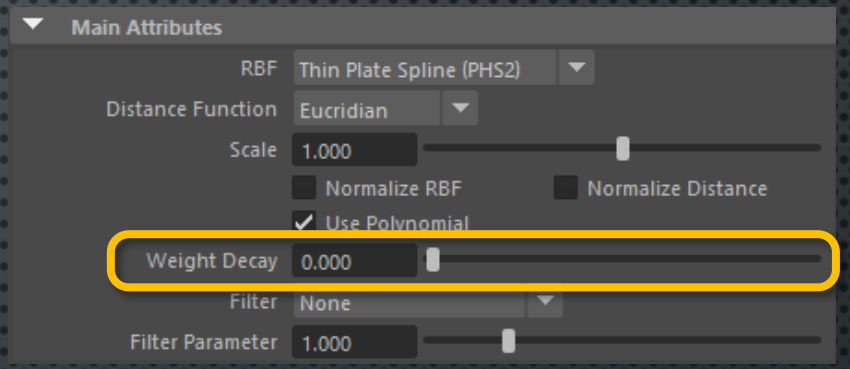
© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

Weight Decay 正則化

過学習（オーバーフィッティング）を抑えて補間を滑らかにする。







ウェイトを求めるのに用いた行列の式に対し、ウェイトが大きくなることにペナルティを課すための項を追加し、そのときの誤差が最小となるようにウェイトを決める。

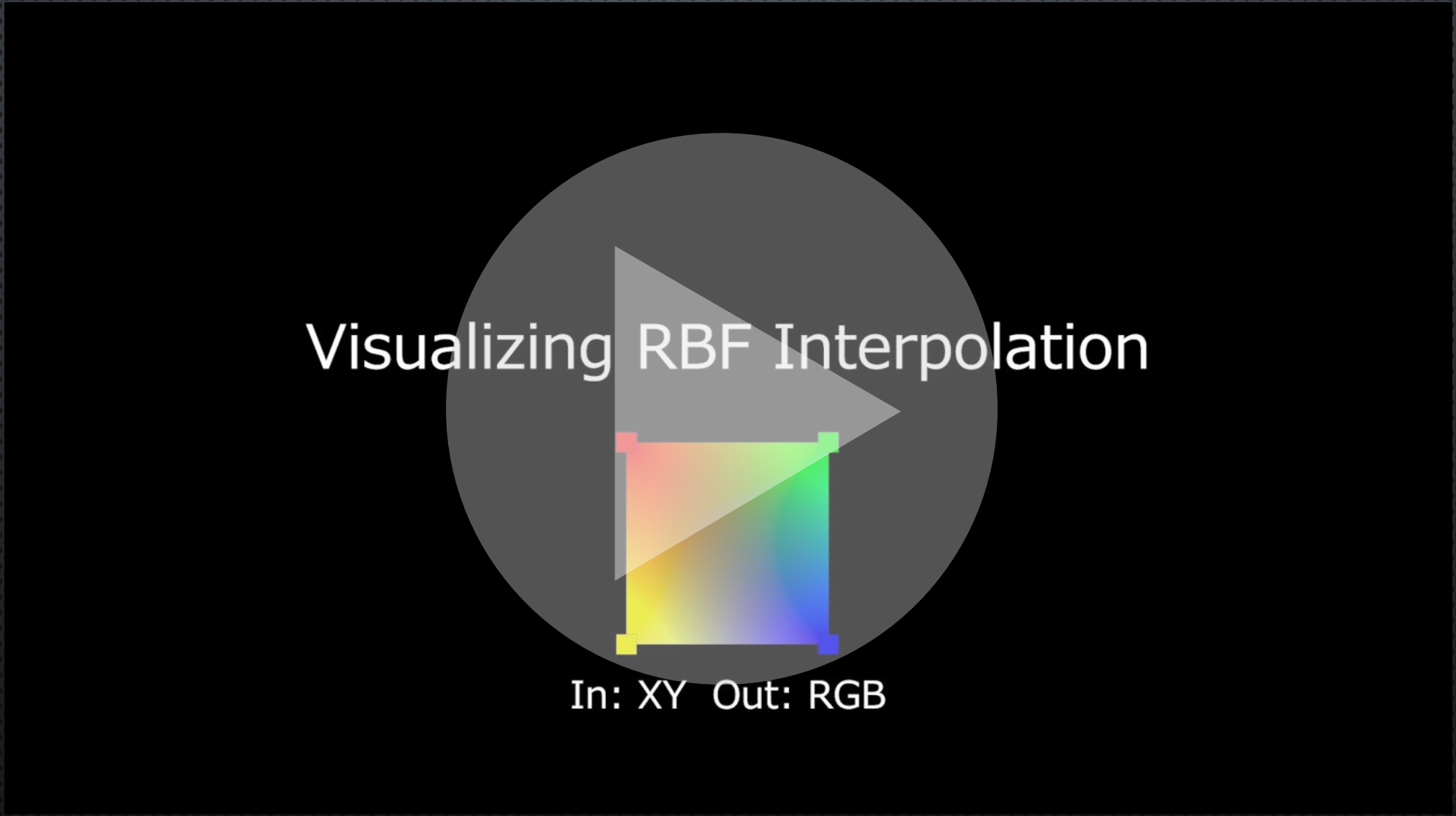
$$\Phi\mathbf{w} = \mathbf{y}$$ （元の線形方程式）

$$\min_{\mathbf{w}}\{\|\Phi\mathbf{w} - \mathbf{y}\|^2 + \lambda\|\mathbf{w}\|^2\}$$ （λ は効果の強さのパラメータ）

$$(\Phi^{\mathrm{T}}\Phi + \lambda\mathbf{I})\mathbf{w} = \Phi^{\mathrm{T}}\mathbf{y}$$ （解くべき線形方程式）

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

デモ：RBF補間の可視化



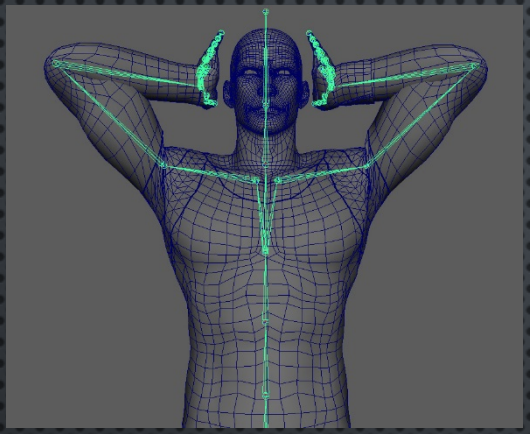
© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

諸機能の実装

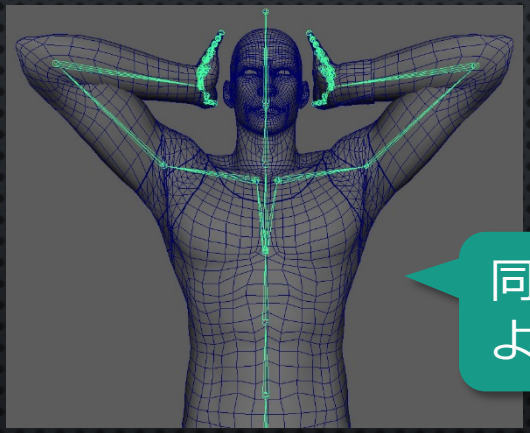
補助骨自動生成

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

補助骨自動生成の概要



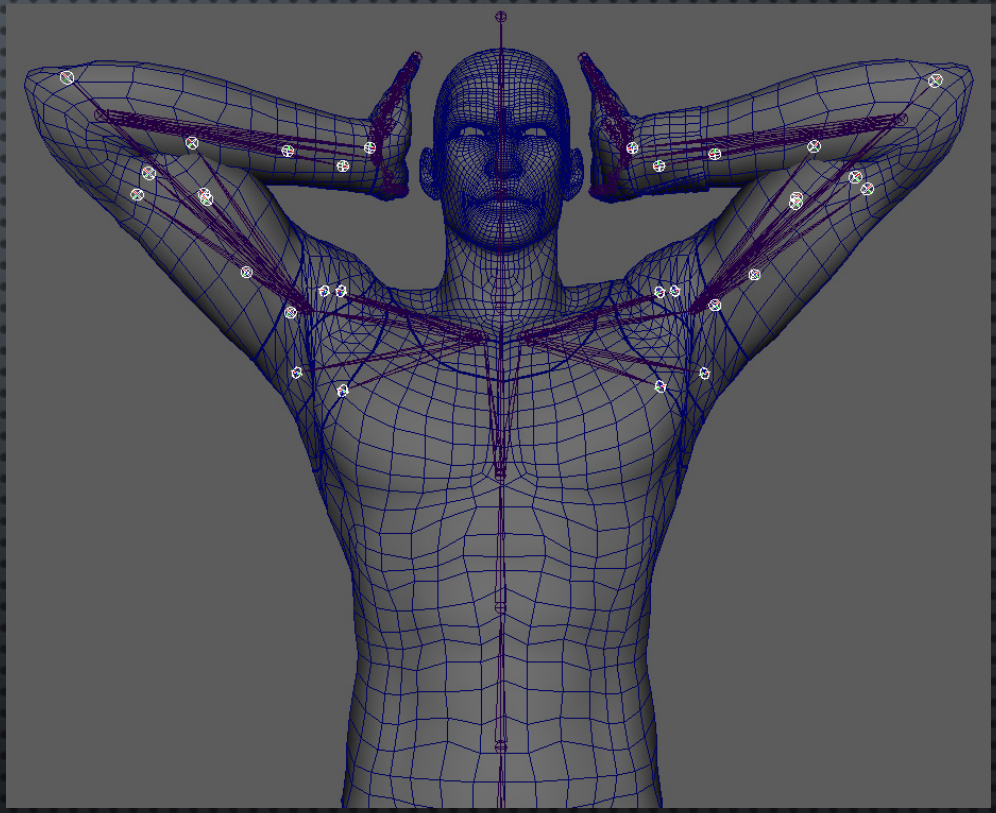
スキン＋スケルトン



例示データ＋スケルトン

同じスケルトンによる高度な変形

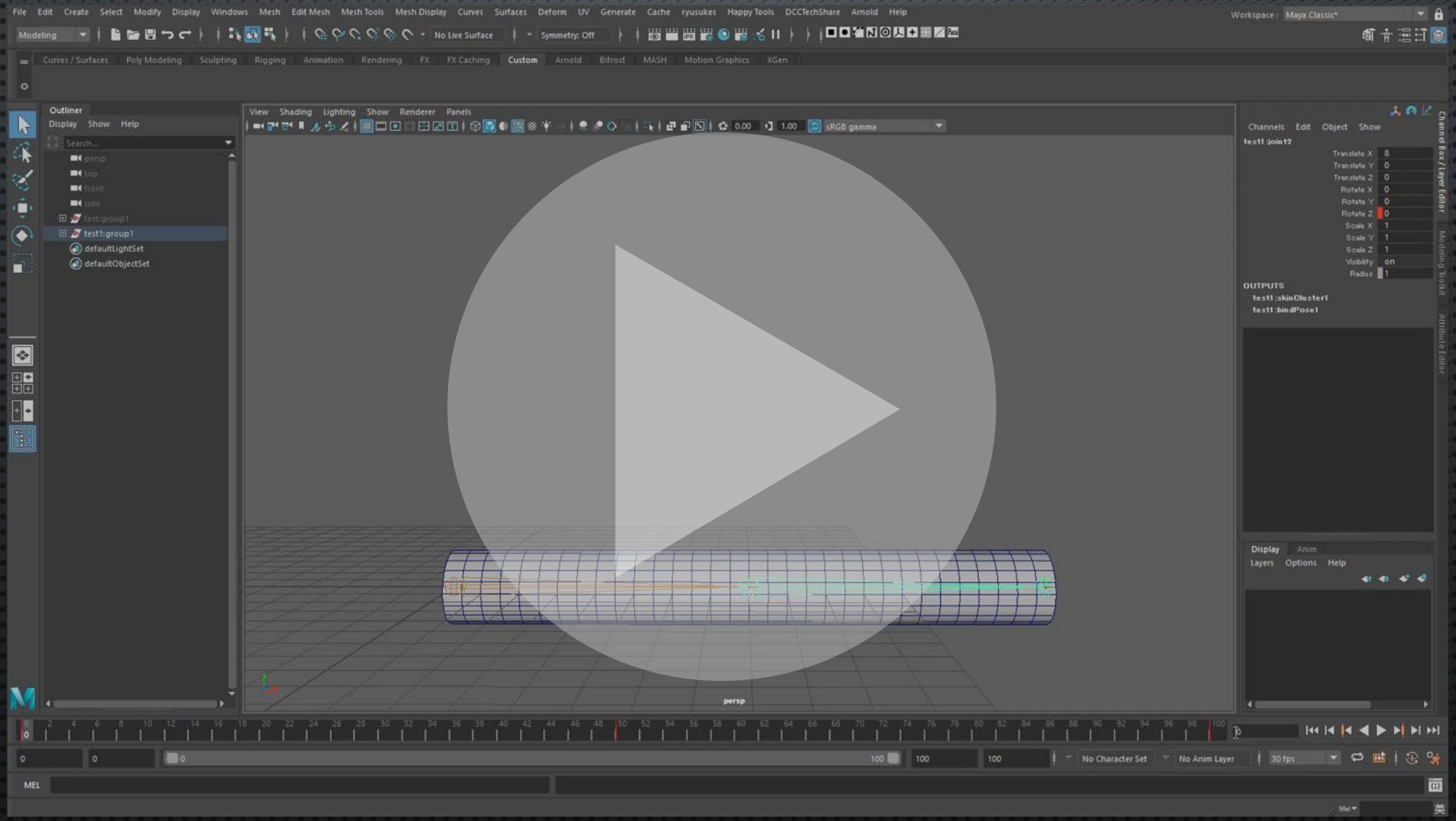
例示データを近似
（補助骨生成）



RBFで駆動する補助骨の追加
スキンウェイトの自動調整

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

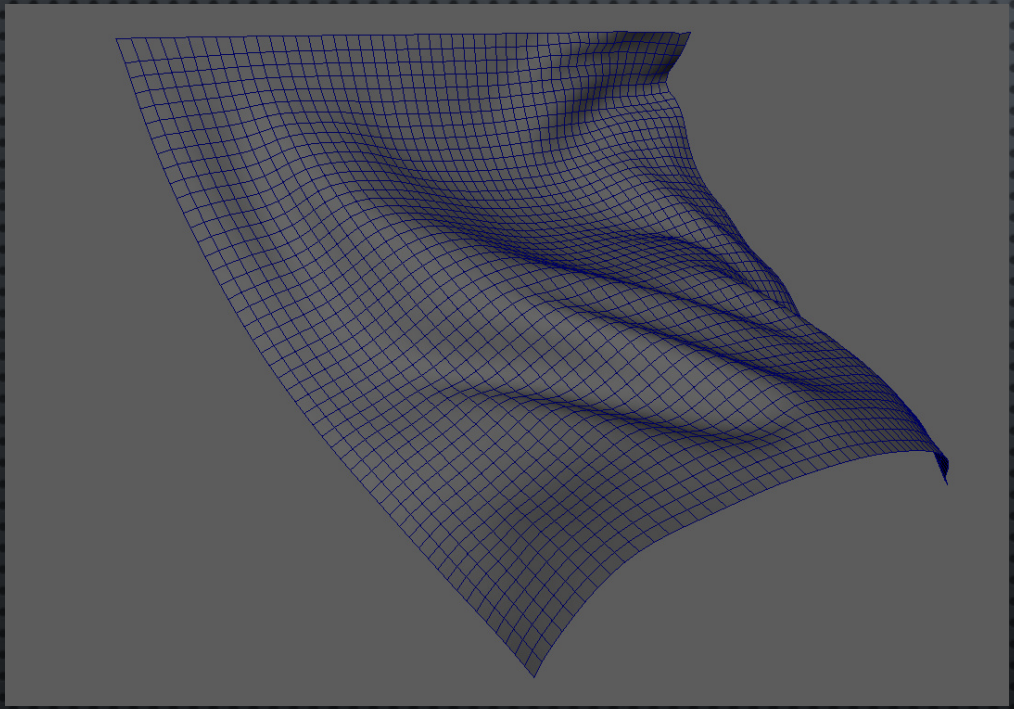
デモ：補助骨自動生成



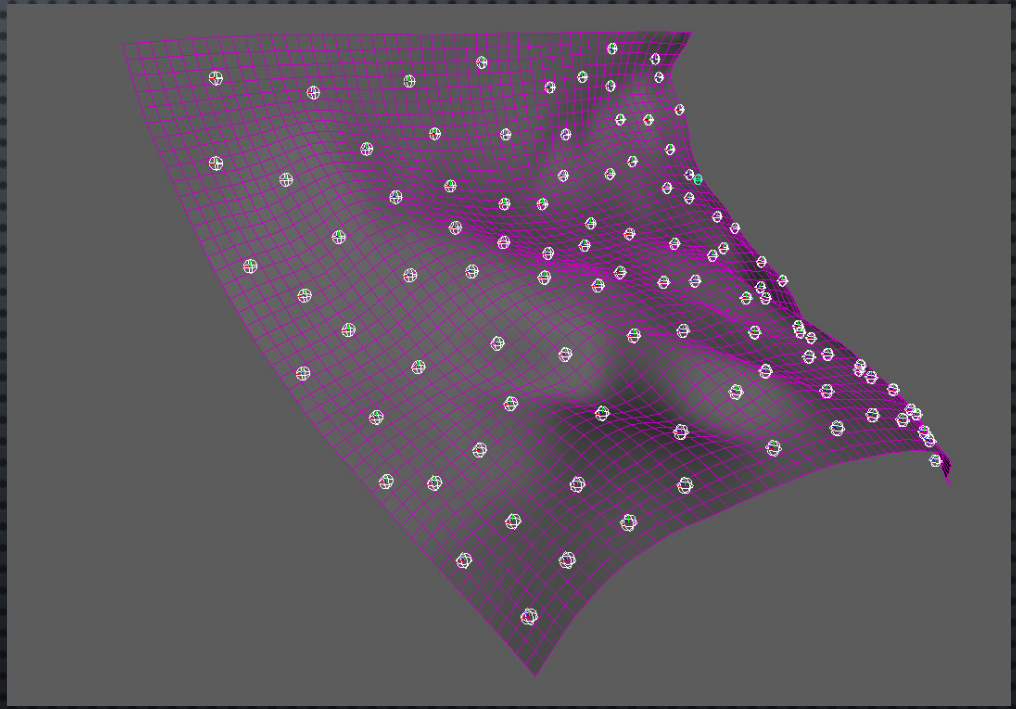
© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

基礎となっている技術

SSDR = Smooth Skinning Decomposition with Rigid Bones
[Le and Deng, 2012]



任意の頂点アニメーション



スキンアニメーション

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

SSDR の概要

J 個の頂点を持つモデルの
N 個の例示デフォーメーションを
D 個のボーンによるスキニングで近似する。

$$\min_{w,\mathbf{S}} \sum_{n=1}^{N} \sum_{j=1}^{J} \left\| \underline{\mathbf{v}_{j,n}} - \underline{\sum_{d=1}^{D} w_{j,d} \mathbf{u}_j \mathbf{S}_{d,n}} \right\|^2$$

頂点の例示位置

スキニング変形結果：
- w スキンウェイト
- $\mathbf{u}$ 頂点の初期位置
- $\mathbf{S}$ ボーンのスキン変換行列

subject to

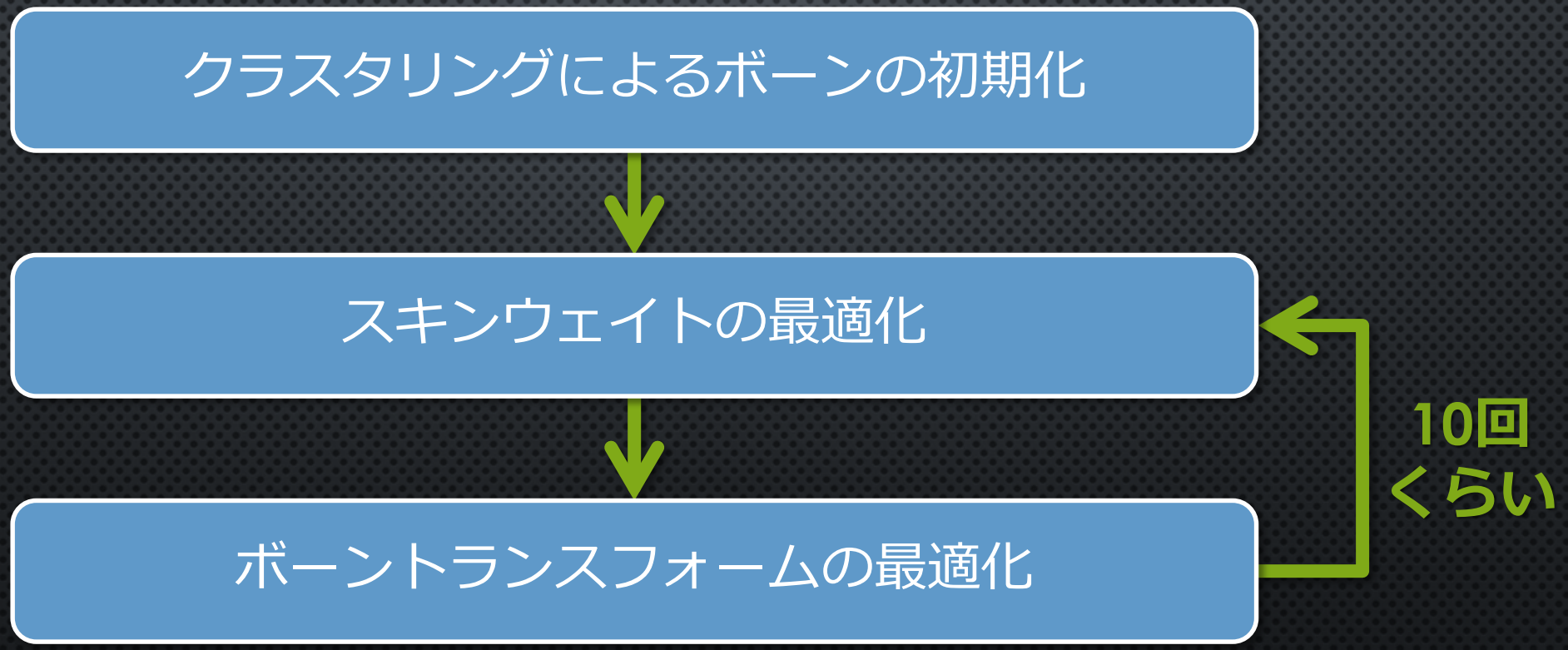
$$w_{j,d} \geq 0, \qquad \sum_{d=1}^{D} w_{j,d} = 1, \qquad |w_j|_0 \leq K$$

- 負のウェイト無し
- 各頂点のウェイトの合計は 1.0
- 各頂点に影響するボーンの最大数は K

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

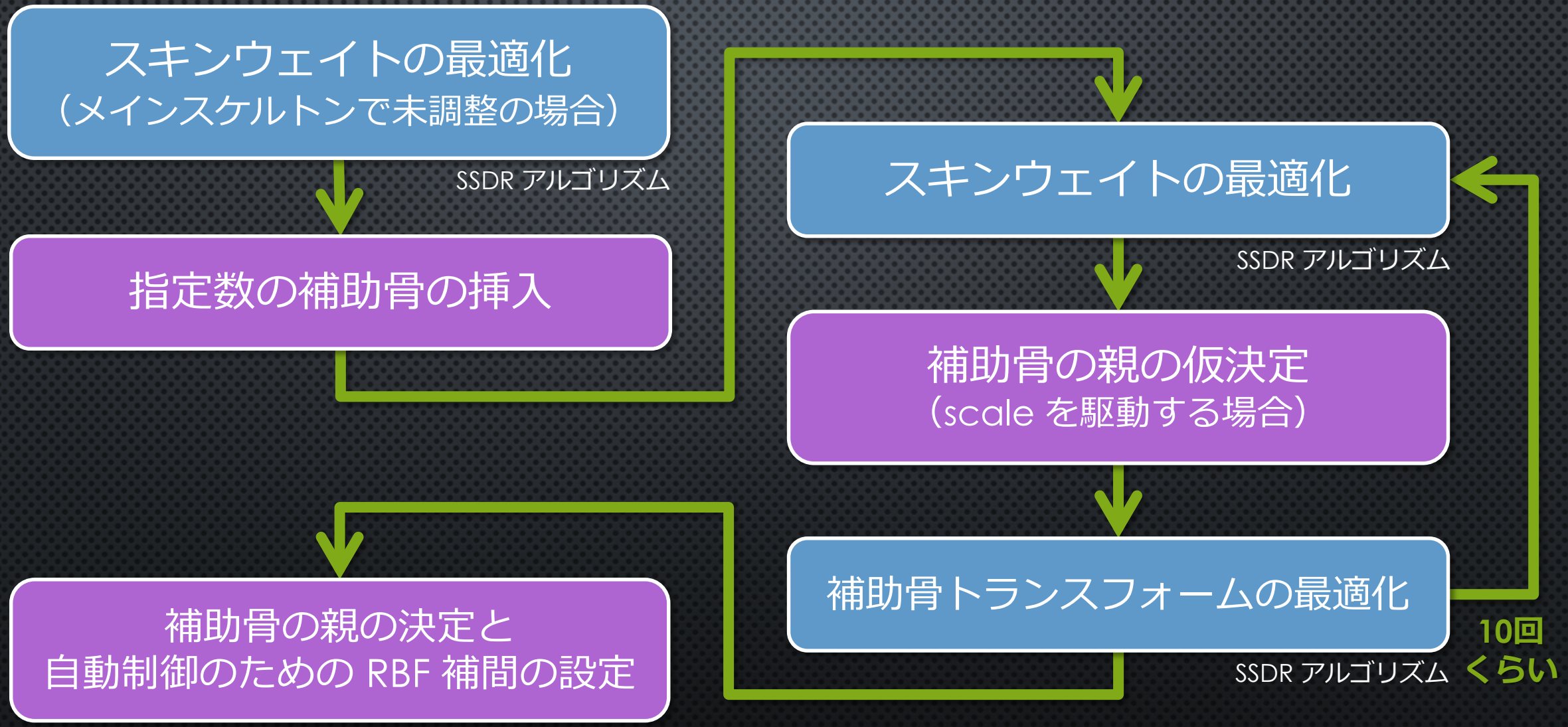
SSDR 処理手順の概要

スキンウェイトとボーントランスフォームを同時に最適化するのは困難なので、交互に最適化して収束させる。



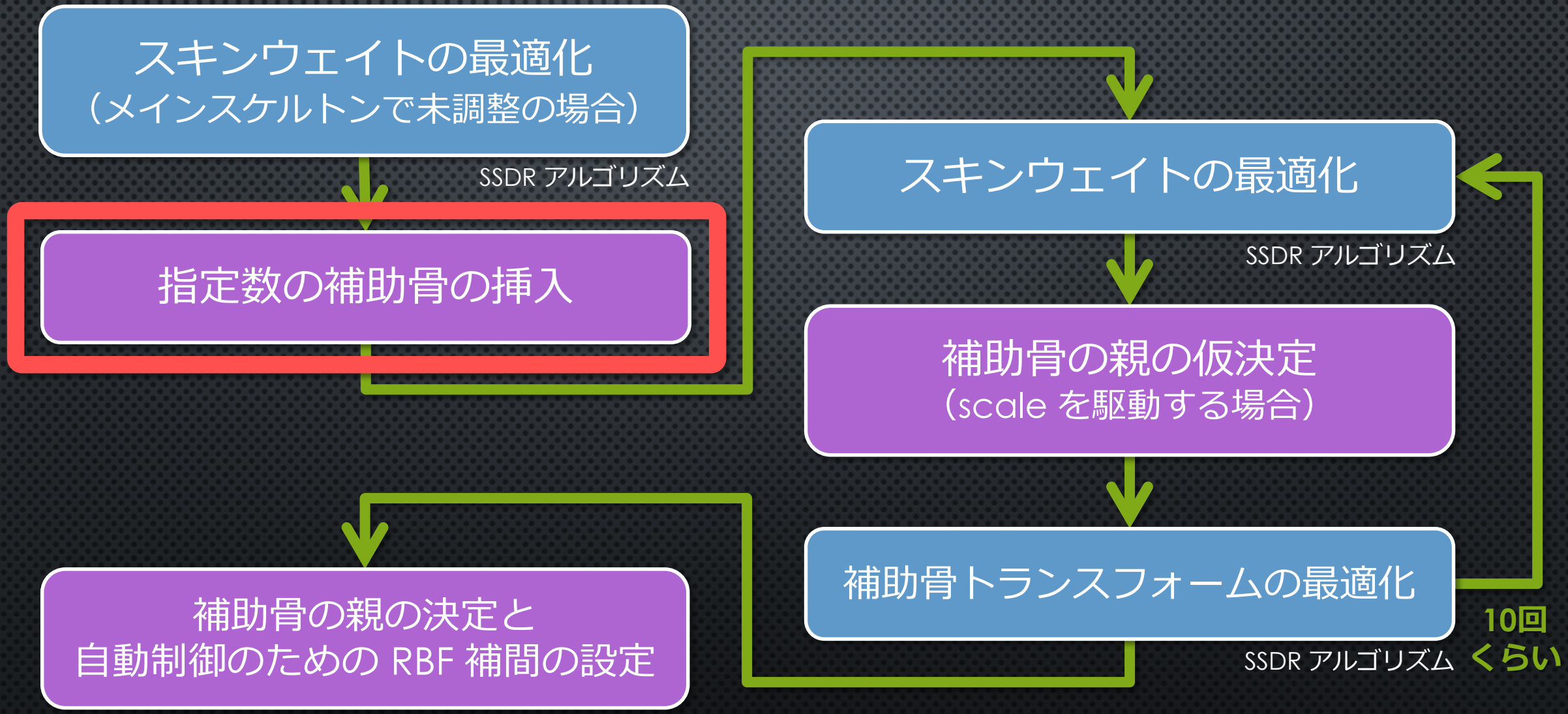
© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

補助骨自動生成の処理手順



© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

補助骨自動生成の処理手順



© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

補助骨の挿入処理

誤差が最大の頂点位置に補助骨を追加

↓

最低限のウェイトの初期化

↓

補助骨トランスフォームの初期化

↓

スキンウェイトの最適化

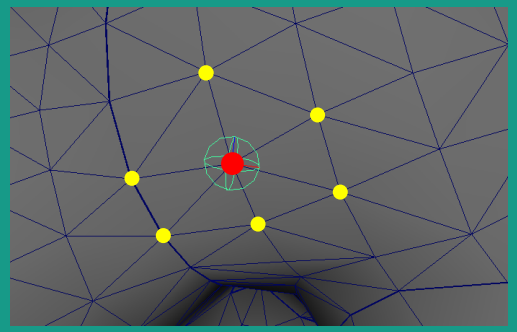
↓

補助骨トランスフォームの最適化

メインスケルトンによるスキニングと例示データを比較し、誤差が最大の頂点を探す。

その頂点と周囲の頂点のみをウェイト 1.0 とし、トランスフォームを初期化する。

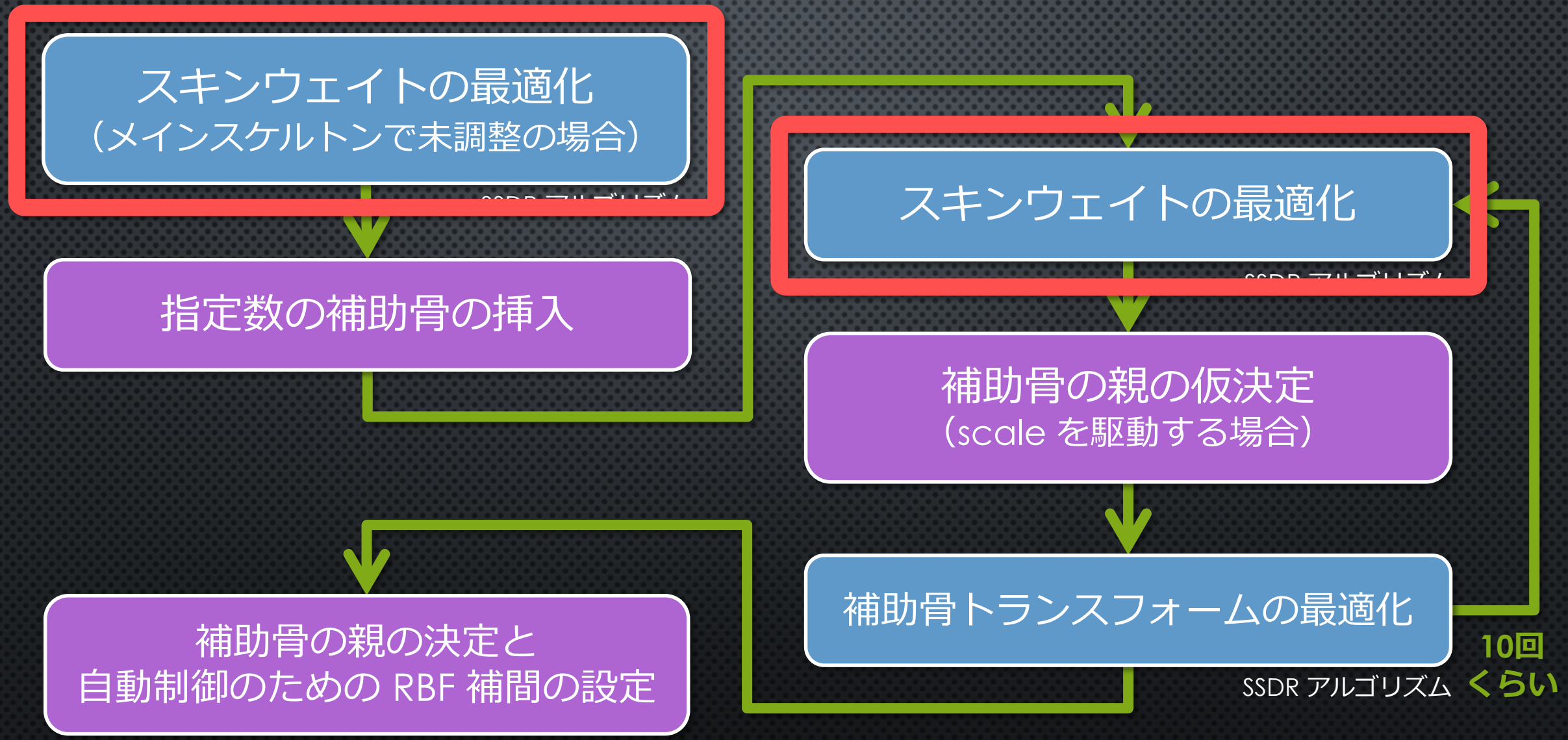
mesh ならば隣の頂点まで。



指定数繰り返し

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

補助骨自動生成の処理手順



© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

スキンウェイトの最適化

最初に示した最適化問題（二次計画問題）の w についてだけ解く。

Computer Graphics Gems JP 2015 の向井さんの記事を参考に実装。

補助骨生成用途では、二次計画問題が安定して解けるように以下を拡張。

頂点の重複例示データの除去

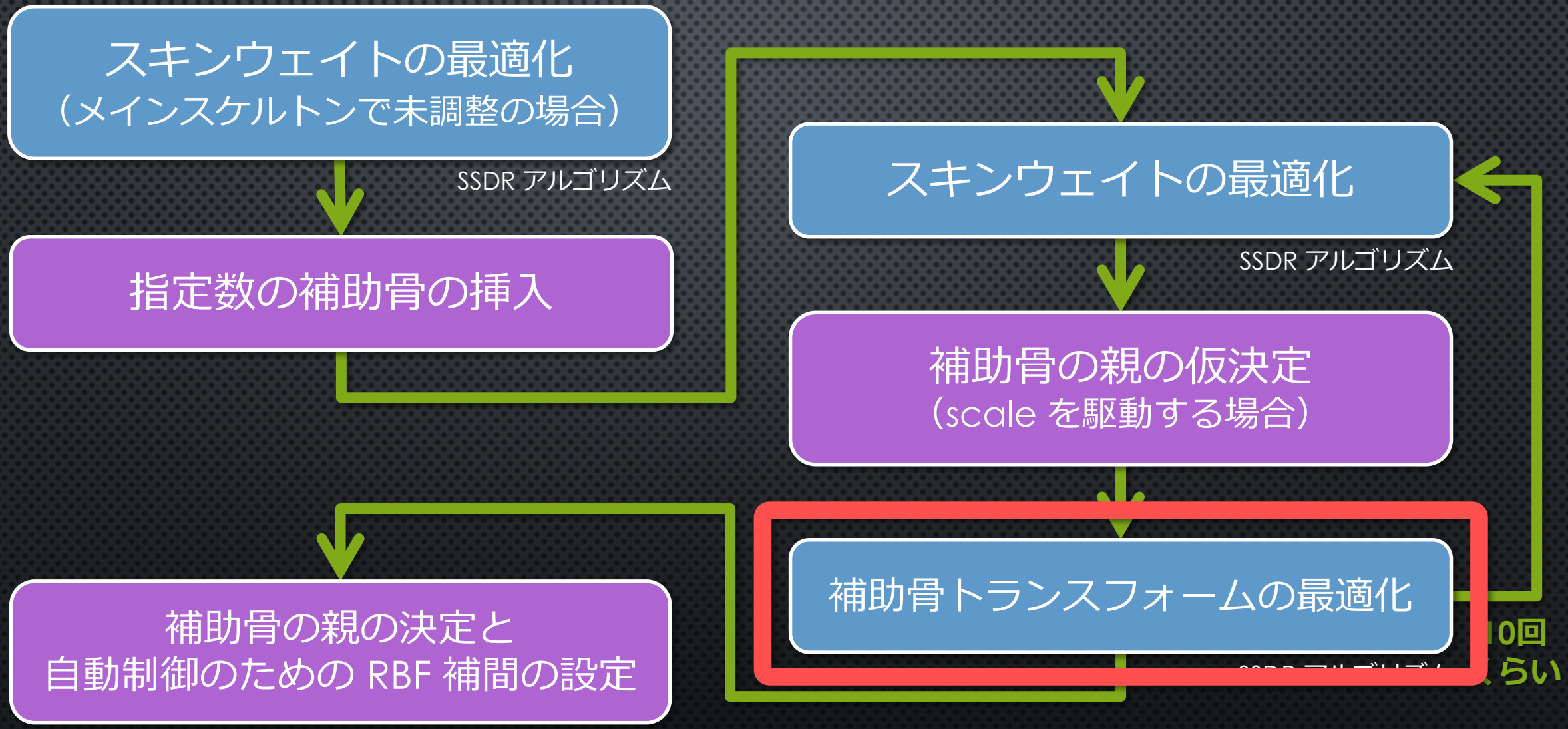
キャラクタの網羅的なポーズを学習させようとするとき、個々の頂点単位でみると動いていないフレームがある。

最適化候補ボーンのグループ化

ある頂点に対して同じ変換結果をもたらすボーンが複数存在する状況がある。それらはグループ化して１つのボーンとして解き、得られた最適化結果のウェイトを元のウェイト配分に従って分け合う処理をする。

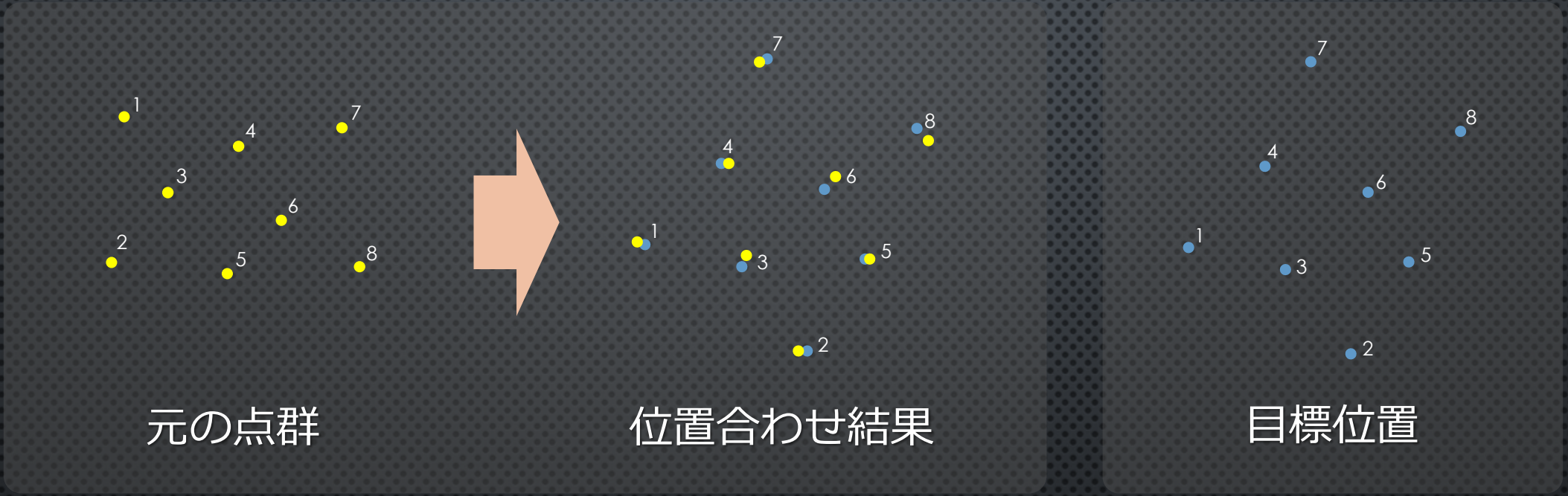
© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

補助骨自動生成の処理手順



© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

トランスフォームの最適化

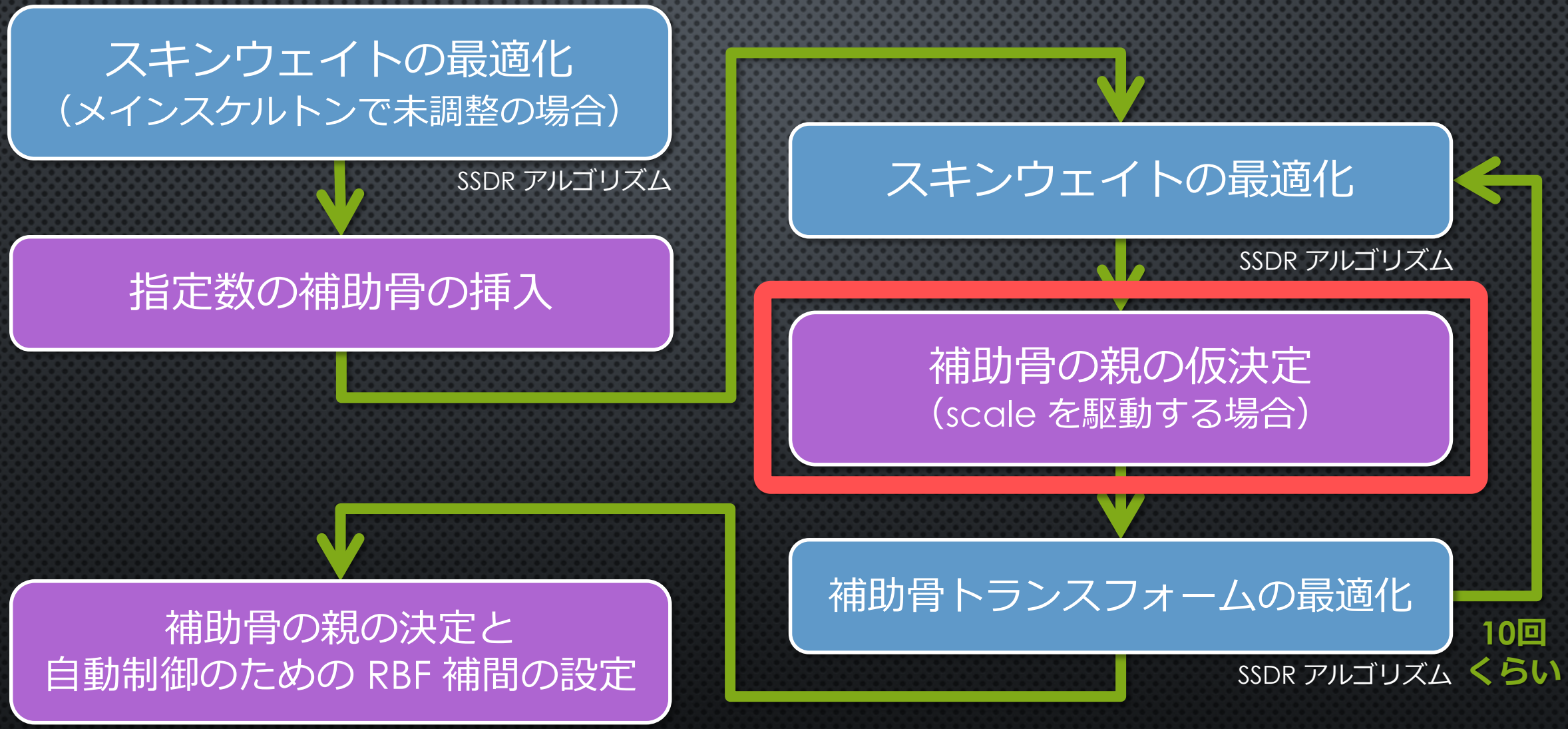


点群を目標にフィットさせる Translation, Rotation, Scaling を求める。

Computer Graphics Gems JP 2015 の向井さんの記事を参考に実装。
[Horn 1987] + Scaling 制御の追加 [Mukai 2018]

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

補助骨自動生成の処理手順



© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

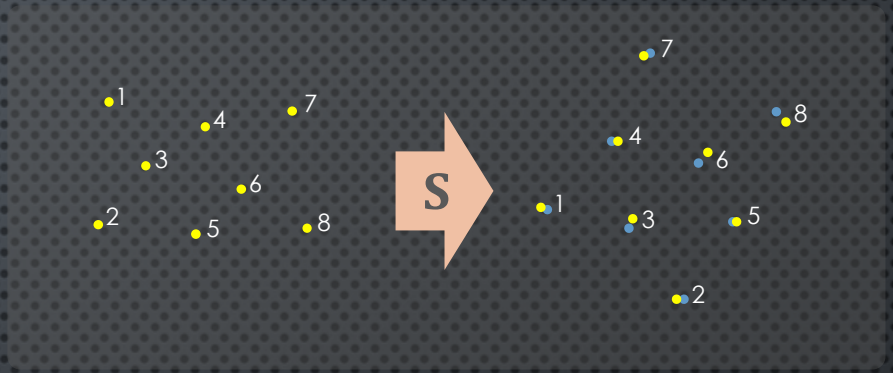
補助骨のスキニング行列と親決定

SSDR で解くスキニング行列 $\boldsymbol{S}$ とは：

$$\mathbf{S} = \underline{\mathbf{B}}^{-1}\,\underline{\mathbf{T}}$$

ワールドトランスフォーム

バインドポーズ（何でも構わない）

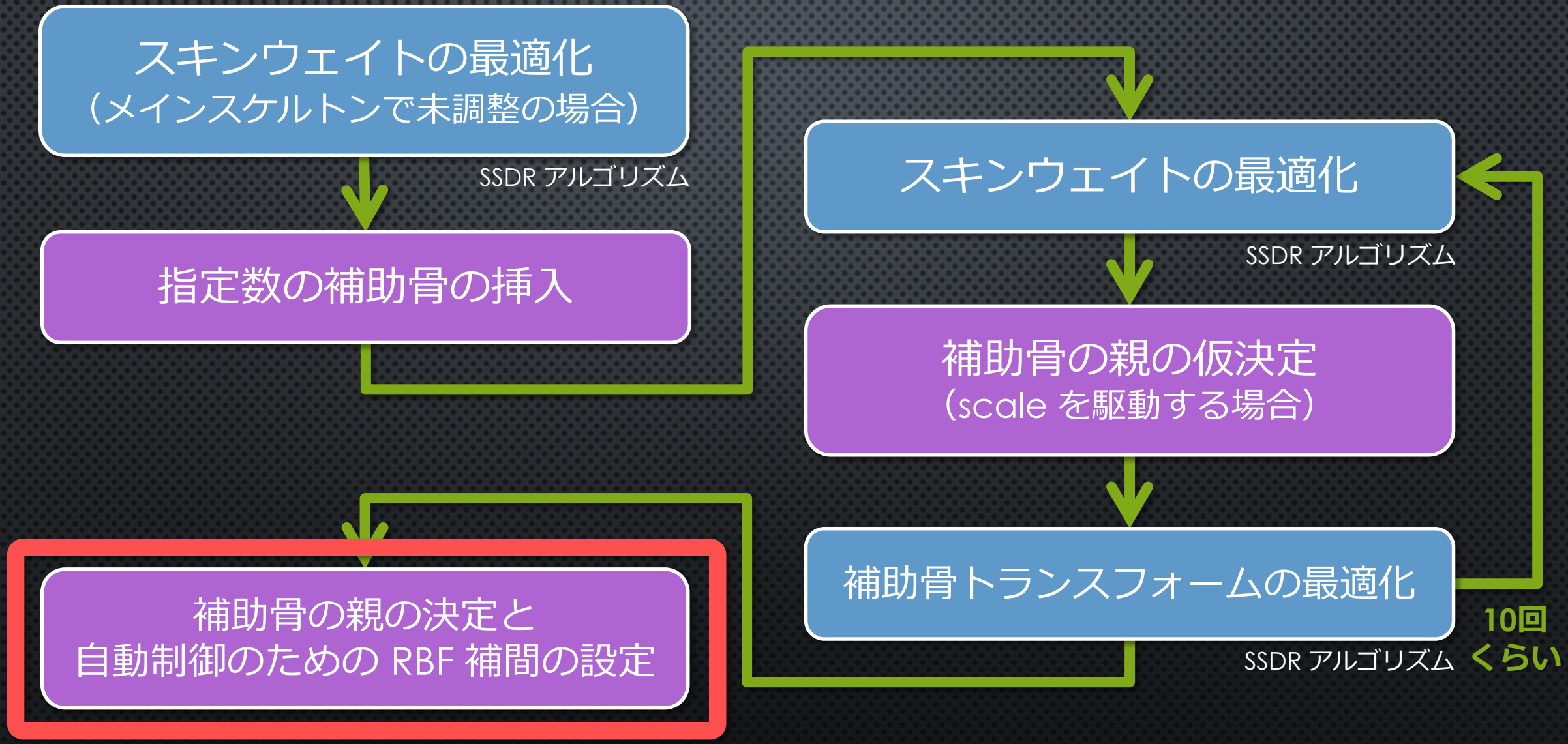


補助骨の場合、安定制御のためにローカル rotate がゼロの状態をニュートラルとしたいため、バインドポーズは親の回転姿勢となる。それを考慮せずに scale を算出すると shear が発生してしまう。

そこで、トランスフォーム最適化の前に 親の仮決定 を行い、スキニング行列ではなくトランスフォーム行列を解くように拡張した。

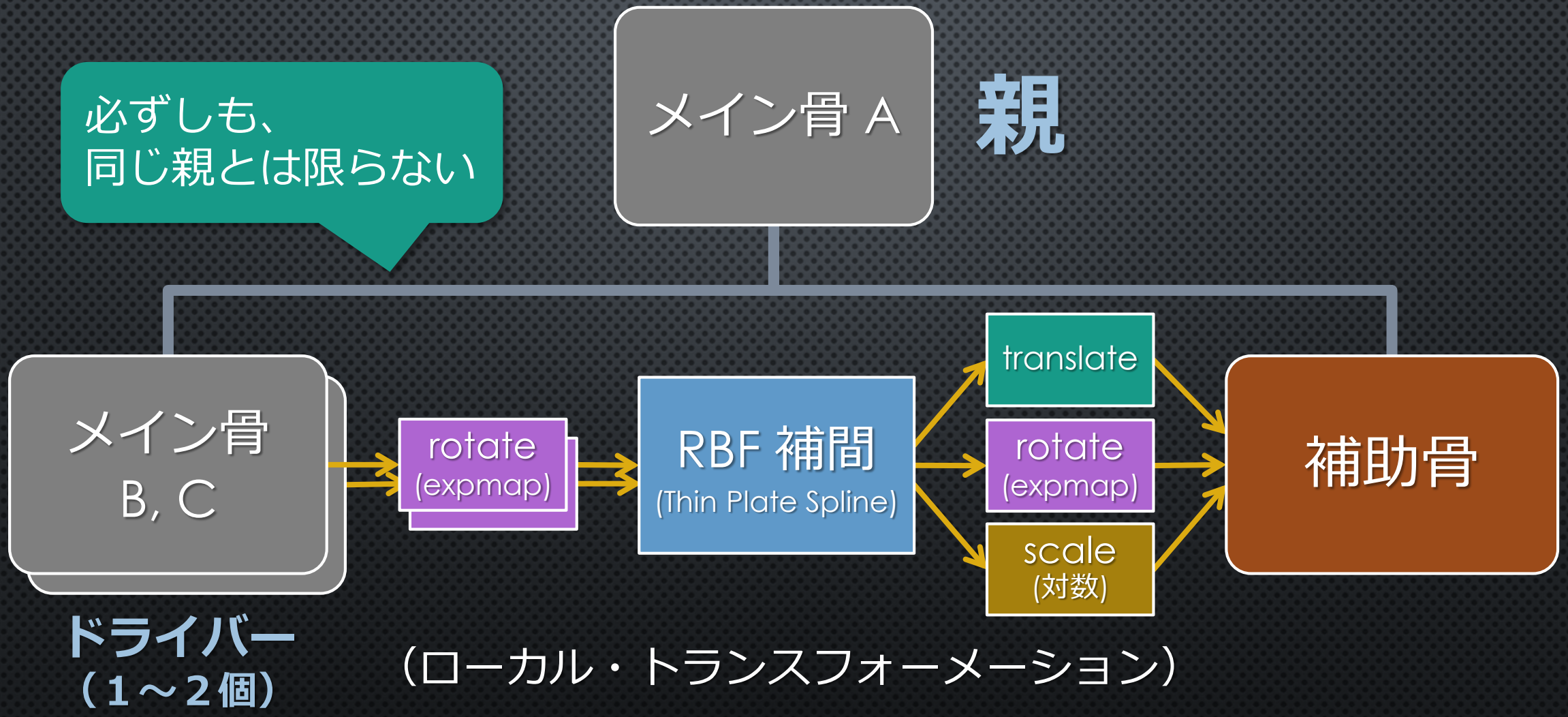
© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

補助骨自動生成の処理手順



© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

KineDriver における補助骨の親とドライバー



© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

補助骨の自動制御の設定

スキンウェイトからドライバー候補を収集

補助骨にバインドされた頂点の補助骨挿入前のスキンウェイトをメイン骨の影響マップと考える。

↓

ドライバー候補から親を決定

親のトランスフォームをそのまま与えたとして最も誤差が小さい親を選ぶ。

↓

残り候補からドライバーを決定し
RBF キーを設定

補助骨の RBF 補間のユニークキーをなるべくカバーできる組み合わせ（１～２個）を選ぶ。

↓

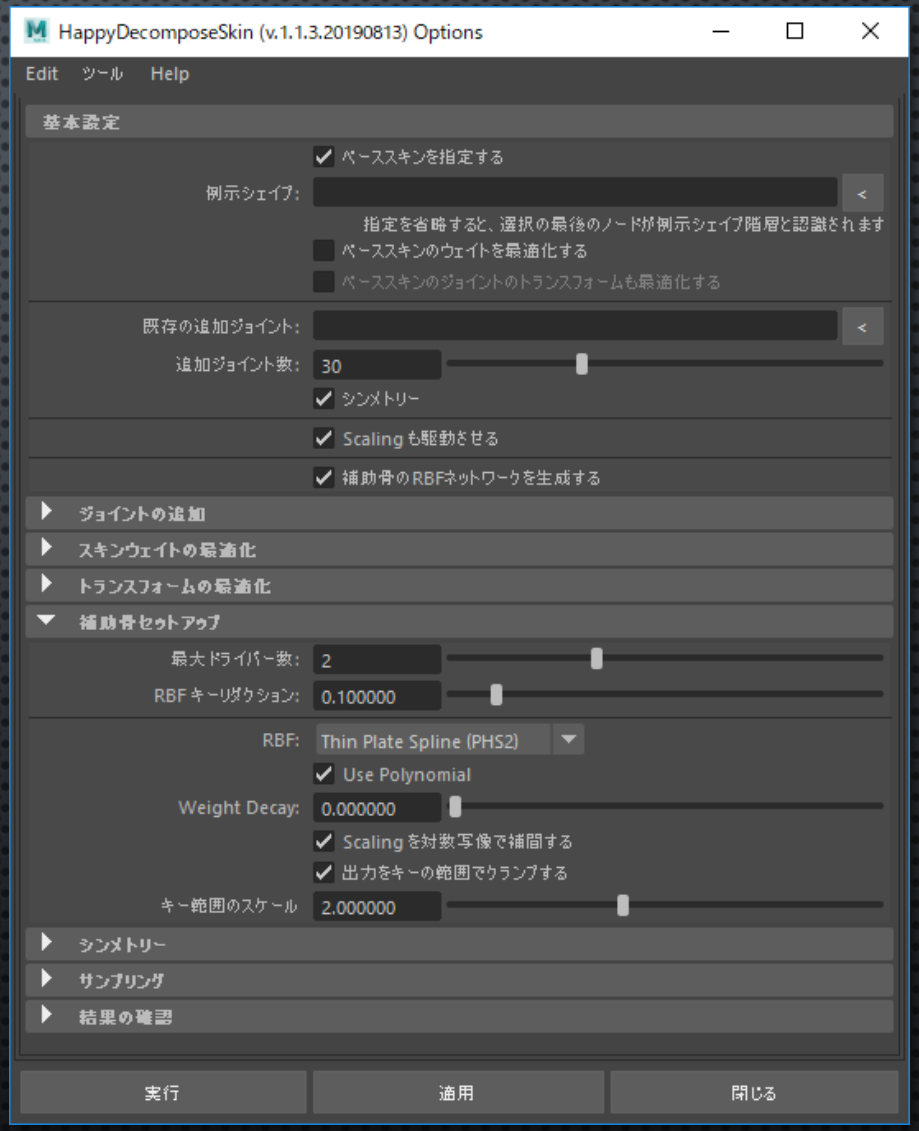
RBF キーのリダクション

キー無しから誤差最大箇所に順次キーを追加していき閾値で終了。

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

その他の拡張

- 高速化の工夫
- シンメトリー補助骨生成
- あらかじめ用意した補助骨の指定
- 例示モーション自動生成（開発中）
- 多用途のスキニング補助ツールとして



© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

参考文献

- Tomohiko Mukai, Sampling-based Rig Conversion into Non-rigid Helper Bones, In Proc. of the ACM on Computer Graphics and Interactive Techniques, Volume 1 Issue 1, (2018), Article 13
- Tomohiko Mukai, Building Helper Bone Rigs from Examples, In Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games 2015, pp.77-84
- 向井 智彦, スキニング分解, Computer Graphics Gems JP 2015 コンピュータグラフィックス技術の最前線, ボーンデジタル, (2015), Chapter 7, pp.139-170
- Binh Huy Le and Zhigang Deng, Smooth skinning decomposition with rigid bones, ACM Transactions on Graphics, Volume 31 Issue 6, (2012), Article 199
- Binh Huy Le and Zhigang Deng, Robust and Accurate Skeletal Rigging from Mesh Sequences, ACM Transactions on Graphics, Volume 33 Issue 4, (2014), Article 84
- Berthold K.P. Horn, Closed-Form Solution of Absolute Orientation using Unit Quaternions, Journal of the Optical Society A, Vol.4, No.4, (1987), pp.629-642
- J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum and T. R. Evans, Reconstruction and representation of 3D objects with radial basis functions. In Proc. of ACM SIGGRAPH 2001, pp.67-76
- Polyharmonic spline (5 April 2019, 14:21 UTC). In Wikipedia: The Free Encyclopedia. Retrieved from https://en.wikipedia.org/wiki/Polyharmonic_spline

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

まとめ

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

まとめ

- KineDriver は社内ですでに多くのプロジェクトに採用
  - 制御しやすい角度分解
  - 簡単でわかりやすい接続
  - 各種DCCツールやゲームエンジンに実装可能な汎用性
- 次世代の需要を見越した拡張
  - 複雑な制御のため、ノードグラフ、エクスプレッションで、自由度を高めて表現の幅を広く
  - 回転、移動、カラーなどの多次元の値を補間可能とする RBF 多次元ドリブンキー
  - シワ、筋肉、PSD などを補助骨に自動変換し、表現水準の向上と作業効率化

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.

ご清聴ありがとうございました。

佐々木　隆典
ryusukes@square-enix.com

髙木　啓太
tkeita@square-enix.com

SQUARE ENIX ADVANCED TECHNOLOGY DIVISION

Maya は オートデスク インコーポレイテッド の商標または登録商標です。
Unreal はエピック・ゲームズ・インコーポレイテッドの商標または登録商標です。
Softimage は オートデスク インコーポレイテッド の商標または登録商標です。
その他、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。

© 2019 SQUARE ENIX CO., LTD. All Rights Reserved.