モジュラーリギングシステム CRAFT と内製ツールの紹介

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

SQUARE ENIX ADVANCED TECHNOLOGY DIVISION

アジェンダ

- CRAFT の概要
- CRAFT リグの使用方法
- CRAFT リギング機能
- CRAFT リグモジュールの紹介
- その他のツールの紹介

CRAFT の概要

CRAFT とは

- Maya® をベースとしたモジュラーリギングシステム。
- 弊社におけるAAAタイトルからモバイルまでのゲーム開発、及びプリレンダー映像作品に至るまで広く利用されている。
- 非公開な内製ツールだが、弊社ゲーム開発における協力会社様にも多くご使用頂いている。

CRAFT の歴史

- 2018年現在 CRAFT 1.9.1　様々な機能追加を経て、現在も開発中
- 2013年12月 CRAFT 1.0　モジュラーリギングシステムの正式版
- 2012年 6 月 CRAFT 0.0　モジュール式でない Biped リグ評価版

過去の発表資料

- FINAL FANTASY XV のキャラクターセットアップワークフロー (2017)

  FF15 の開発で CRAFT がどのように利用されたかの話を含む。
  https://spark.adobe.com/page/1zsaZLEiCfYtP/

- スケールを使ったリグのはなし (2016)

  Computer Animation Open Course – Rig and Tools –

  CRAFT の scale 対応の経験に基づくリギング技術トーク。
  http://www.jp.square-enix.com/tech/library/pdf/RigAndTools_ScaleRig.pdf

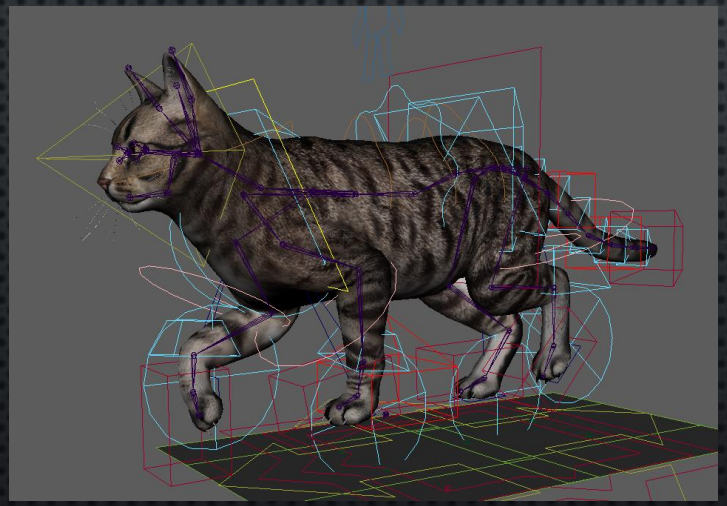
- モジュラーリグシステムのアーキテクチャ (2015)

  CEDEC 2015

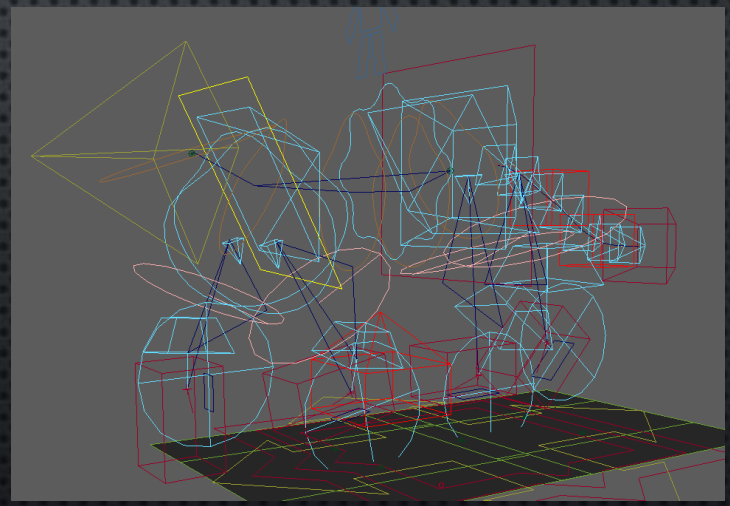
  CRAFT のアーキテクチャの解説。
  http://www.jp.square-enix.com/tech/library/pdf/CEDEC2015_CRAFT.pdf

ツールとしての位置付け

- コントロールリグ作成とそれを用いたアニメーション作成にフォーカスしたツール。
- イメージとしては HumanIK® に近く、デフォーマー等は含まない。



リグ全体
（Deformation + Skeleton + Controller）



コントロールリグ
（CRAFT で主に取り扱うもの）

CRAFT の特徴

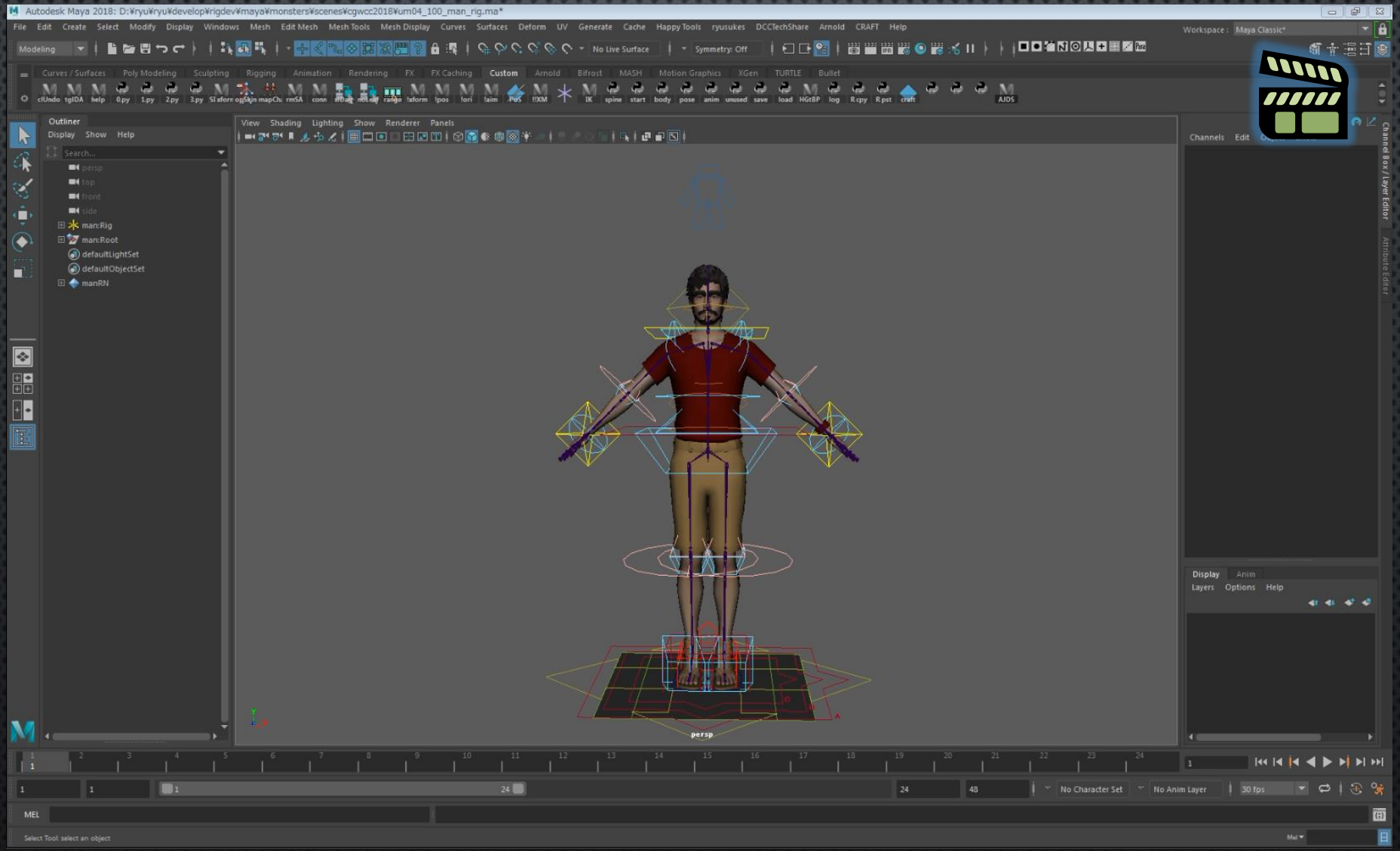
- モジュラーリギングシステム
  - 部品を組み合わせて自由にリグを構築。
  - 既存のあらゆるスケルトンに対応。
- ポーズやアニメーションの転送機能
  - スケルトンのアニメーションをコントロールリグに転送できる。
  - この仕組みに基づき、リターゲットにも対応（モジュールによって対応状況が異なり、現在のところ、完全に対応しているのは Biped のみ）。

CRAFT リグの使用方法

リグの使い方はこんな感じ

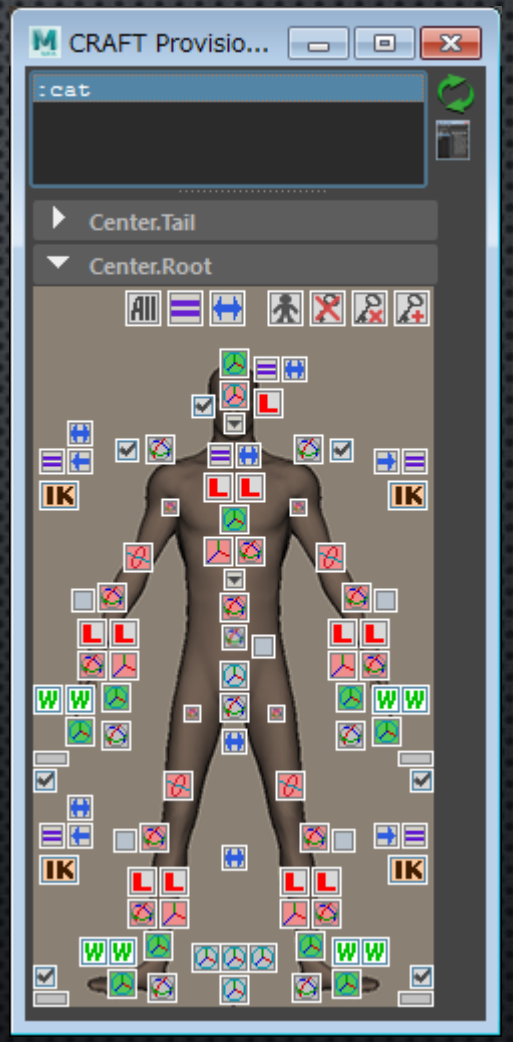
- 標準リグでは、コントローラは全てロケータになっていて、ビュー上で選択できる。
- ロックされていない箇所は自由に動かせる。ピボットなども。
- 各種スイッチ等の調整アトリビュートを集約したロケータが頭上にある。
  - IK/FK, Local/World の切り替え。
  - 自動肩やフロアコンタクトなどの機能の on/off 。
- コントローラは全て管理されており、Rig Explorer でアクセス出来る。アイコン化されていないスイッチノードにアクセスする手段にもなる。

リグの使い方はこんな感じ

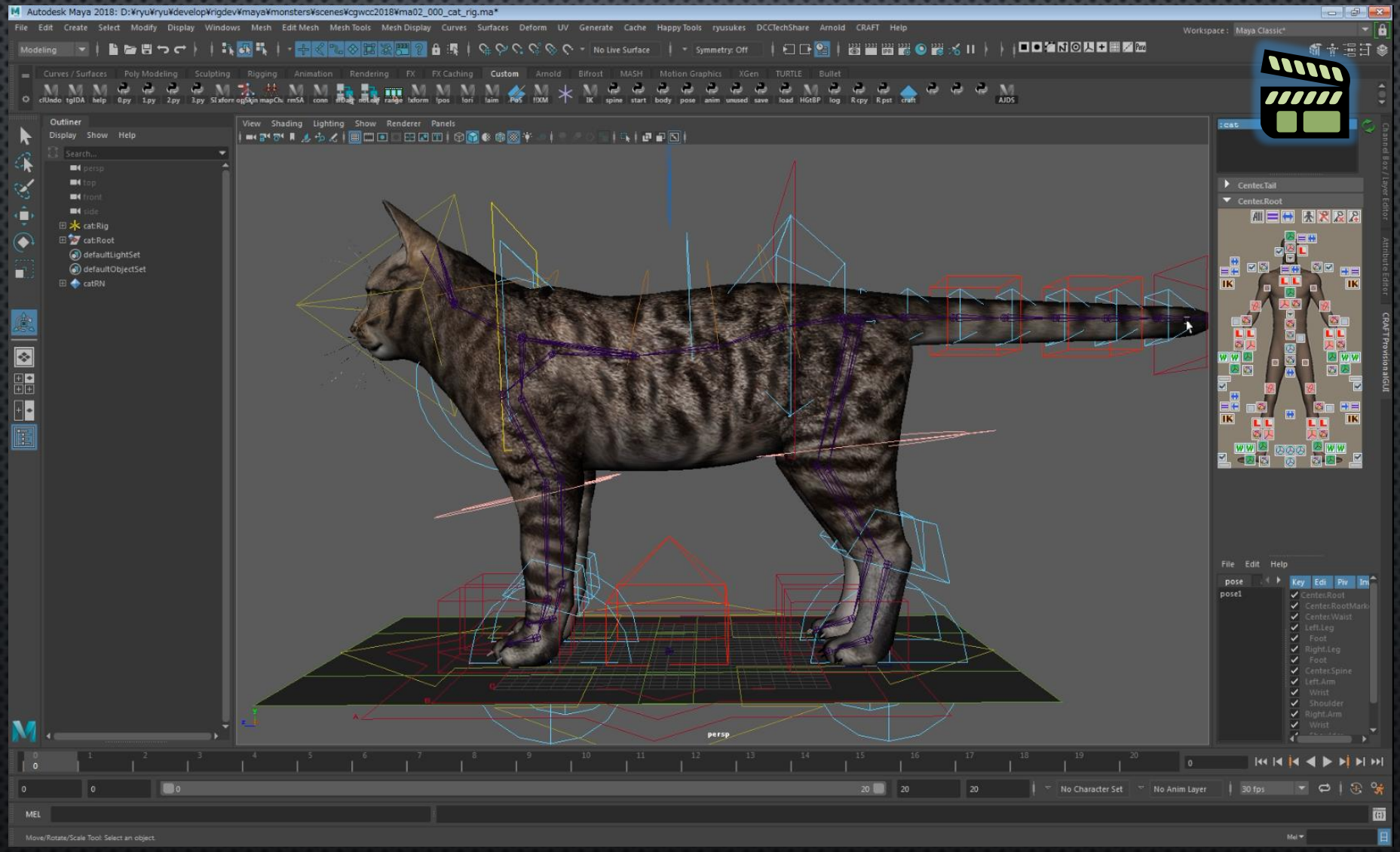


仮のピッカー GUI

- 四肢を持つキャラなら
  Biped でも Quadruped でも扱える。
- 各種切り替えスイッチ。
- フィットとミラー。
- 四肢以外の部位は追加フレームに簡易ボタンが表示される。
- より柔軟な GUI の開発は将来の課題。

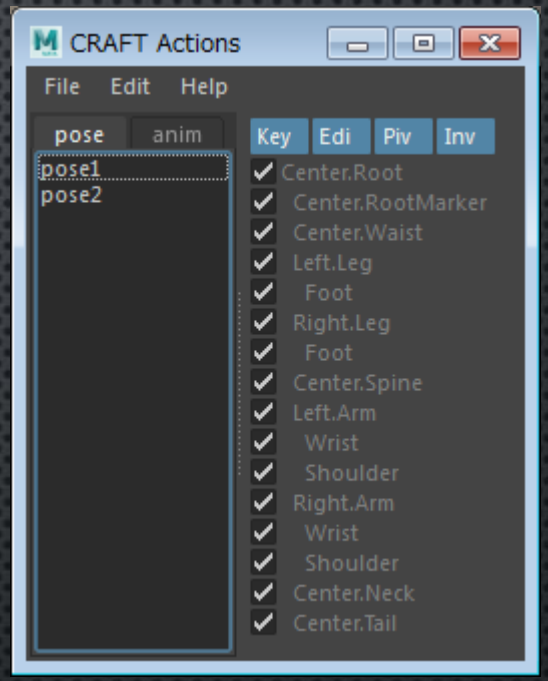


仮のピッカー GUI

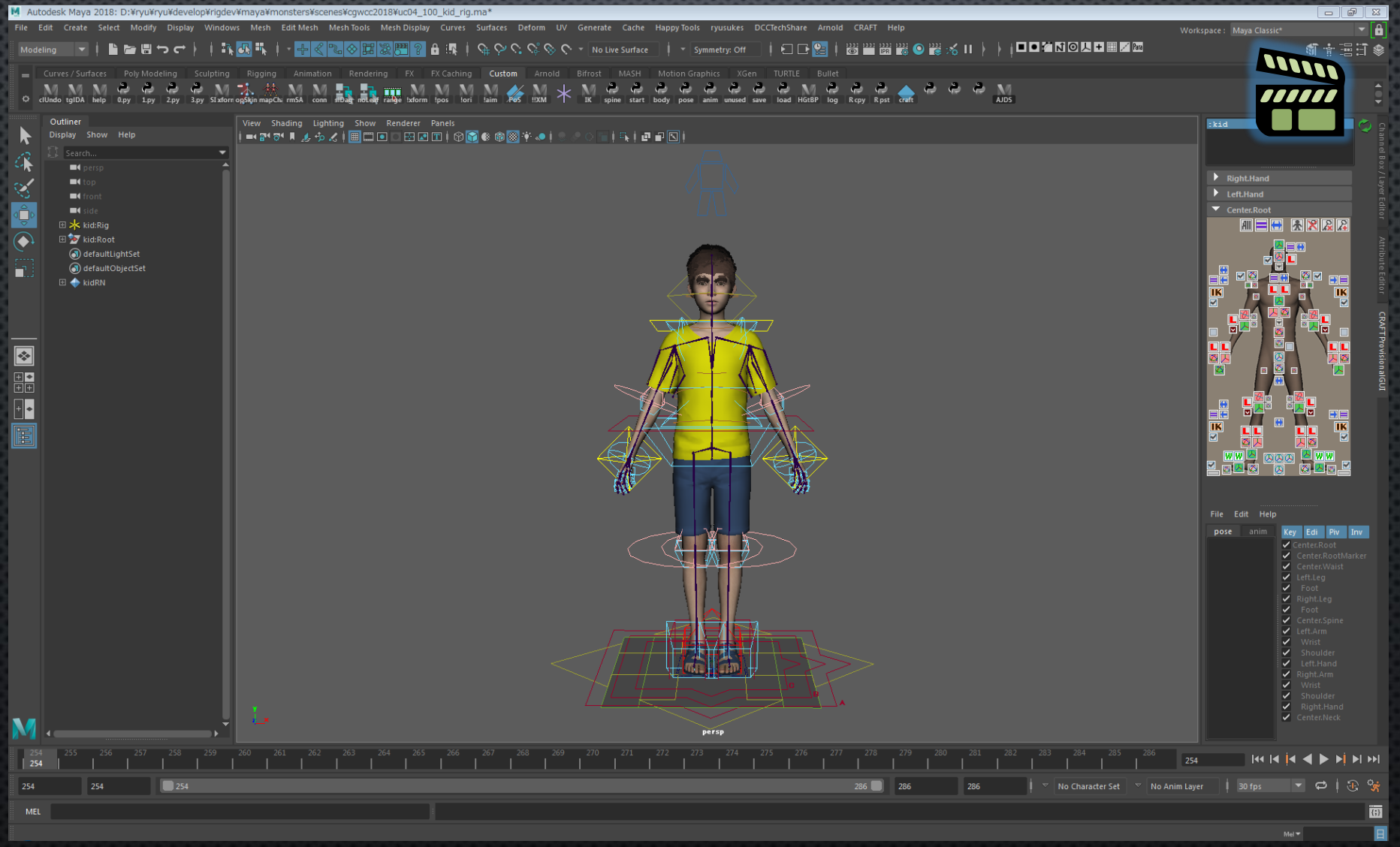


Action Manager

- ポーズとアニメーションのセーブとロード。
- セーブもロードも、部位ごとに on/off 。
- カレントシーン内に埋め込みで保持される。
- ファイルの読み書きもサポート。
- より高機能な GUI の開発は将来の課題。



Action Manager



Convert 機能

- World で付けたアニメーションを Local に変換したい。
- アニメーションをミラーしたい。
- スケルトンのアニメーションをコントロールリグに転送したい。
- アニメーションのリターゲットをしたい。
- （GUI の フィット や ミラー のボタンも実は？）

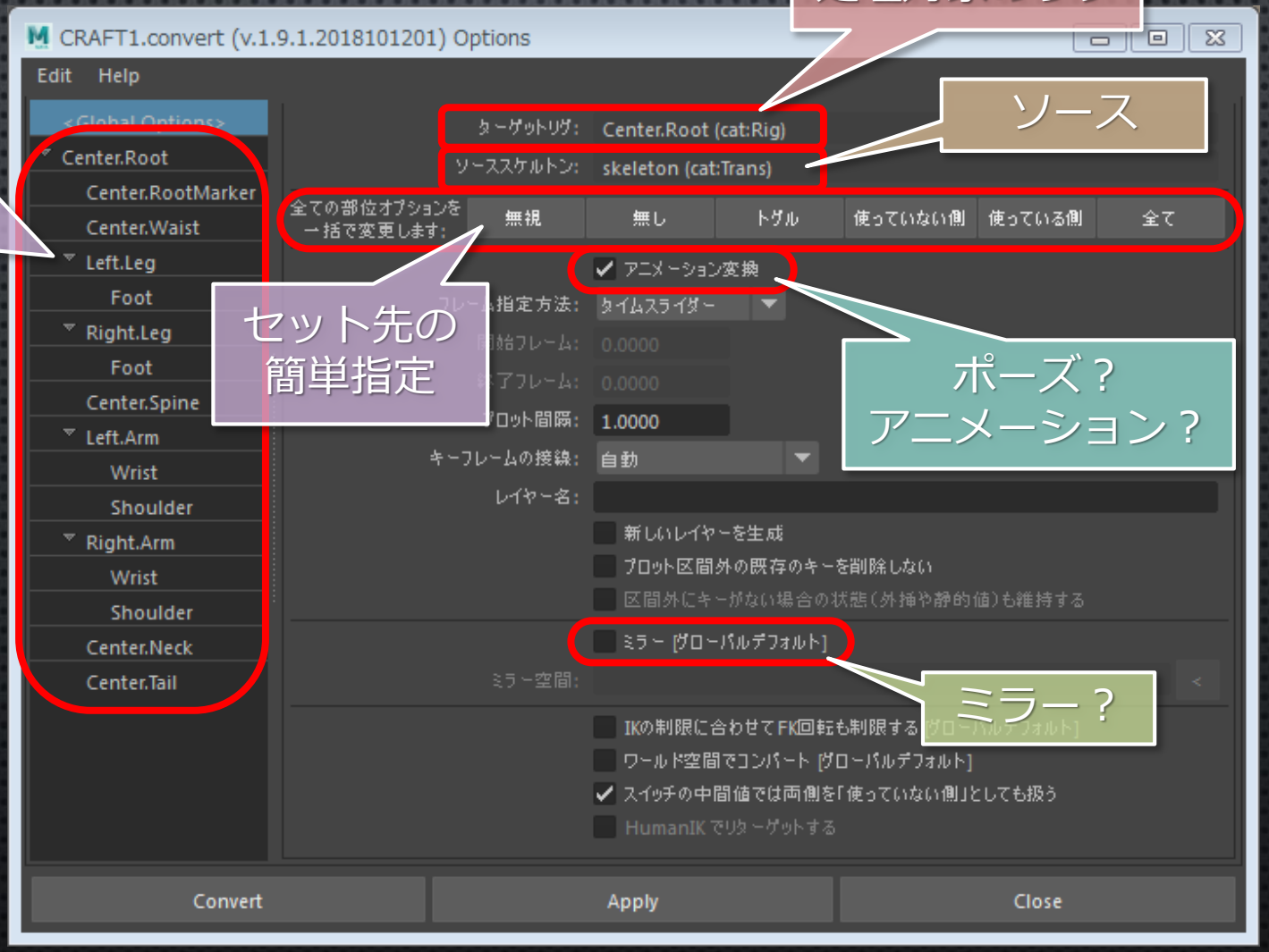
すべて Convert 機能でサポートされる

どういうことなのか？

全ては以下の違いでしかない。

- 何がソース？
  - コントロールリグ自身 … Local/World や IK/FK の変換など。
  - 自身のスケルトン … スケルトンからコントロールリグへの転送。
  - 他のスケルトン … リターゲット。
- そして
  - ポーズ or アニメーション？
  - ミラー？
  - 値のセット先のコントローラは？

CONVERT オプション



値のセット先の指定

- 部位ごとに細かく指定する場合

  例えば、Local to World 変換なら World コントローラがセット先。

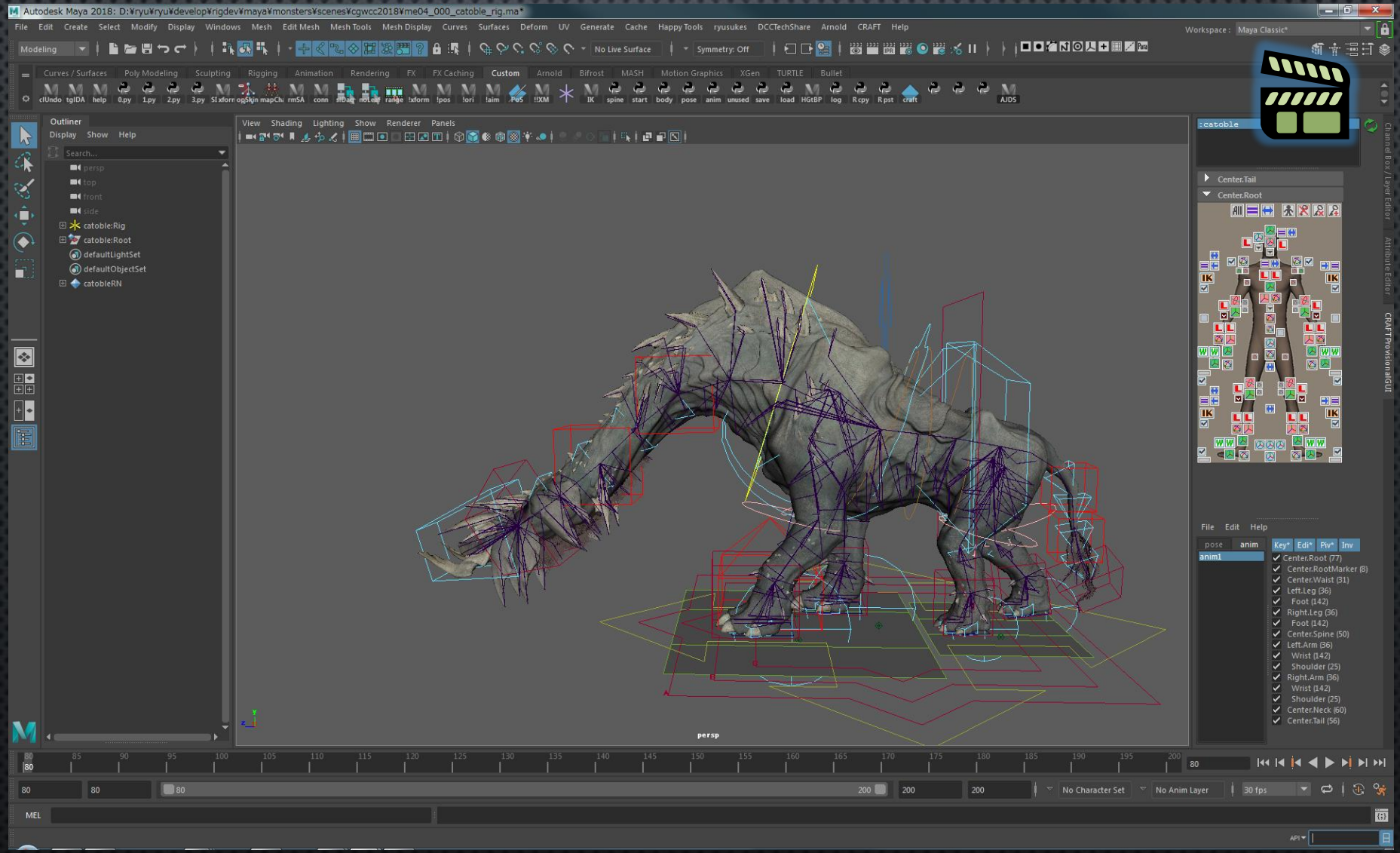
- 簡単に指定する場合
  - 使っている側
  - 使っていない側
  - トグル（共用部分の設定を切り替え）

指定の使い分け

- Local/World や IK/FK の変換の場合
  - コントローラが別々の場合、「使っていない側」を指定。
    ちなみに、GUI の「フィット」ボタンは、
    「使っていない側」への「ポーズ」コンバート。
  - コントローラが共用の場合、「トグル」を指定。
- その他の場合（スケルトンからリグへの転送など）

  「使っている側」か「すべて」を指定。

CONVERT 機能

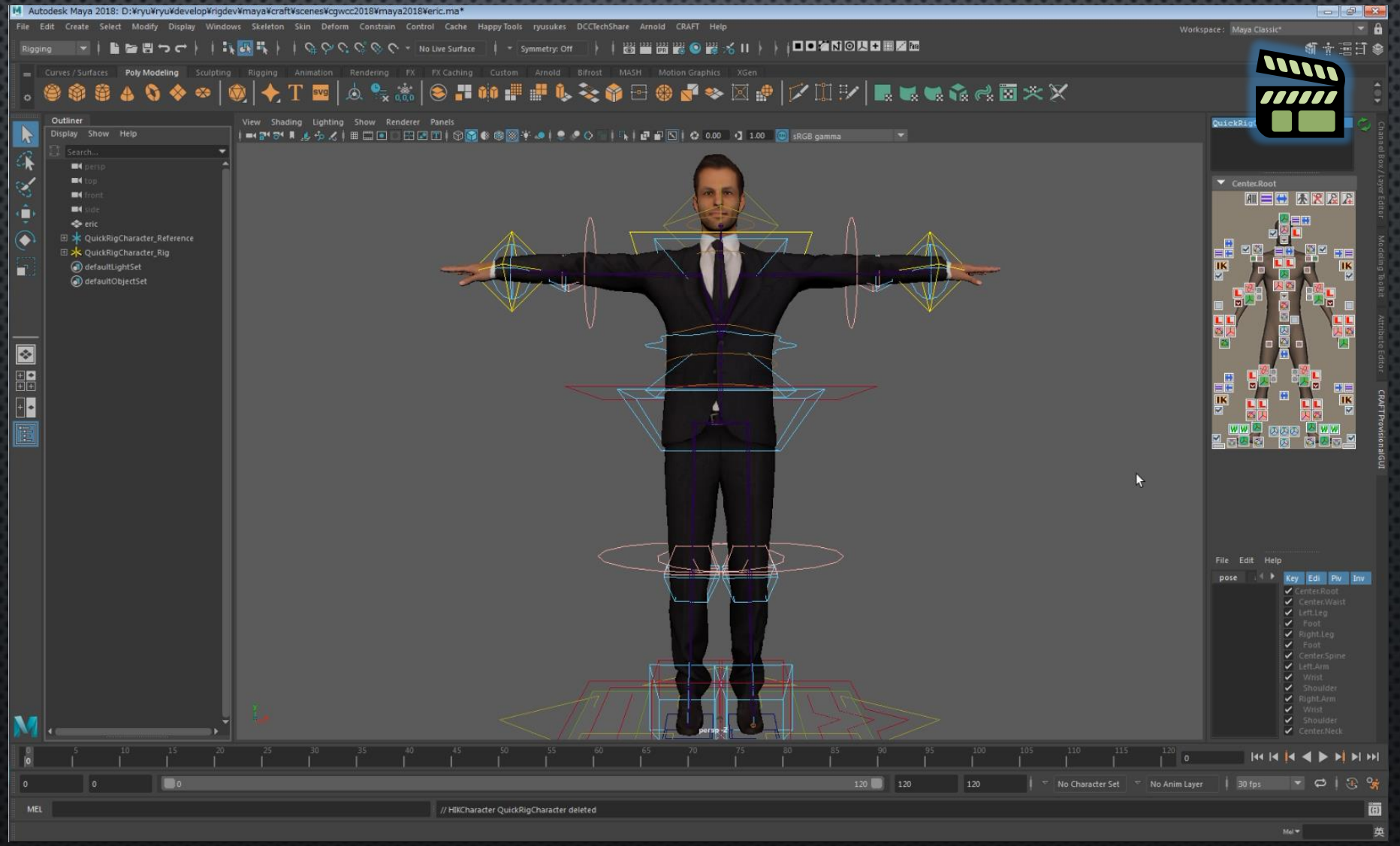


CRAFT リギング機能

Quick Rig ツールで超簡単リグ作成

- CRAFT は HumanIK® と連携できる。
- Maya® の Quick Rig ツールと相性が良い。

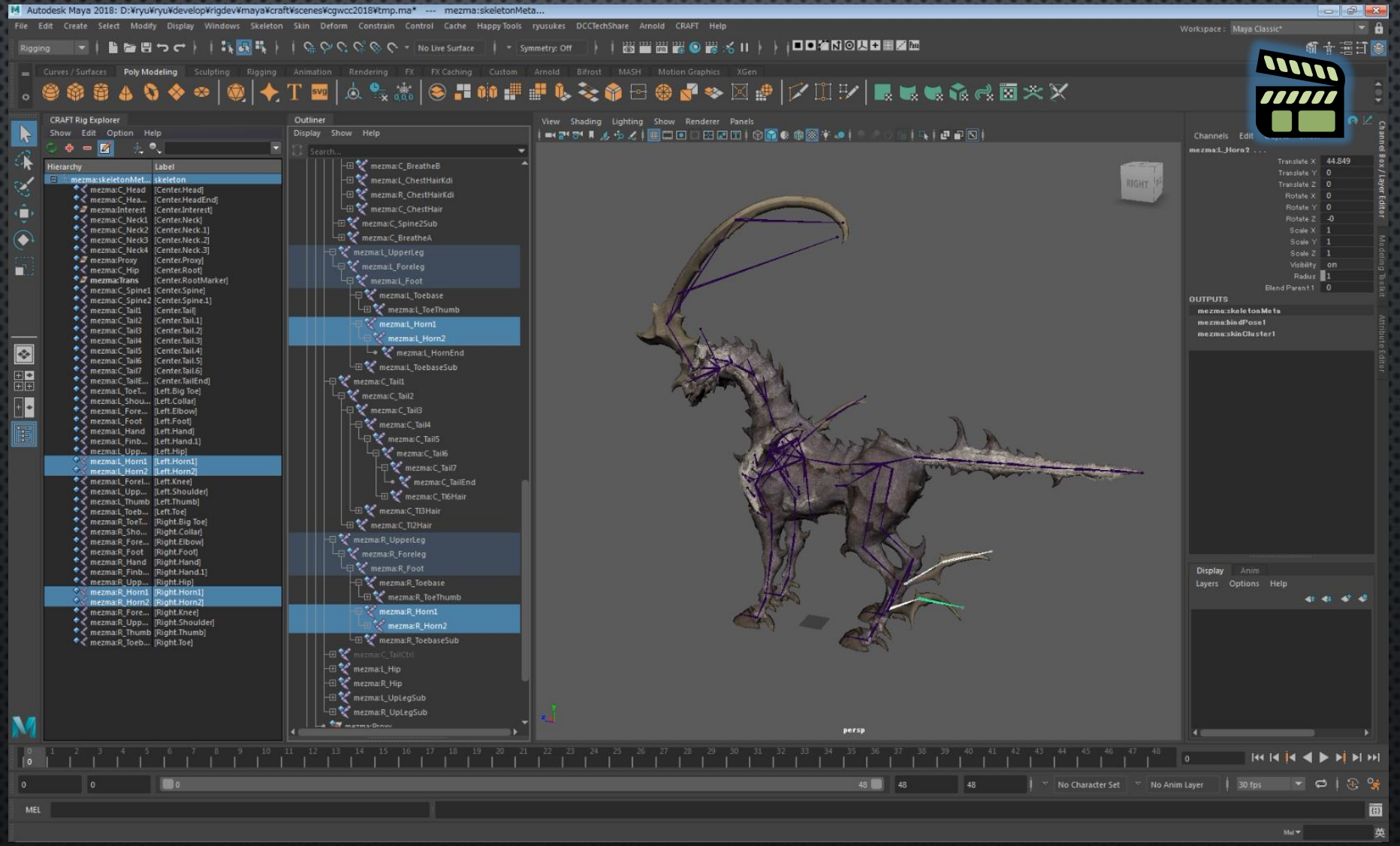
Quick Rig ツールで超簡単リグ作成



キャラクタライズ

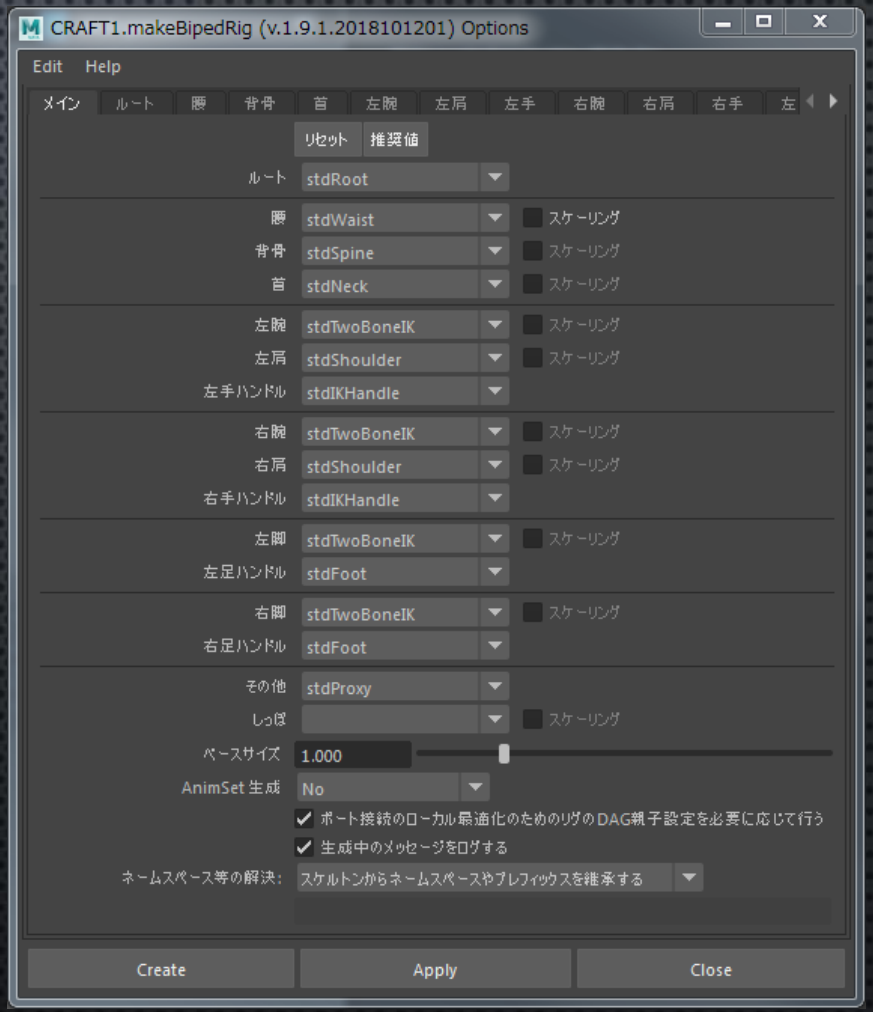
- スケルトンを CRAFT に認識させる作業。
- メタノードに、各関節のラベルと初期ポーズを登録。
- 通常、プリセットテーブルによって、一発で行う。
- 追加のジョイントは Rig Explorer で登録。
- 未知のスケルトンは、簡単なスクリプトを書いて一発で登録。

キャラクタライズ

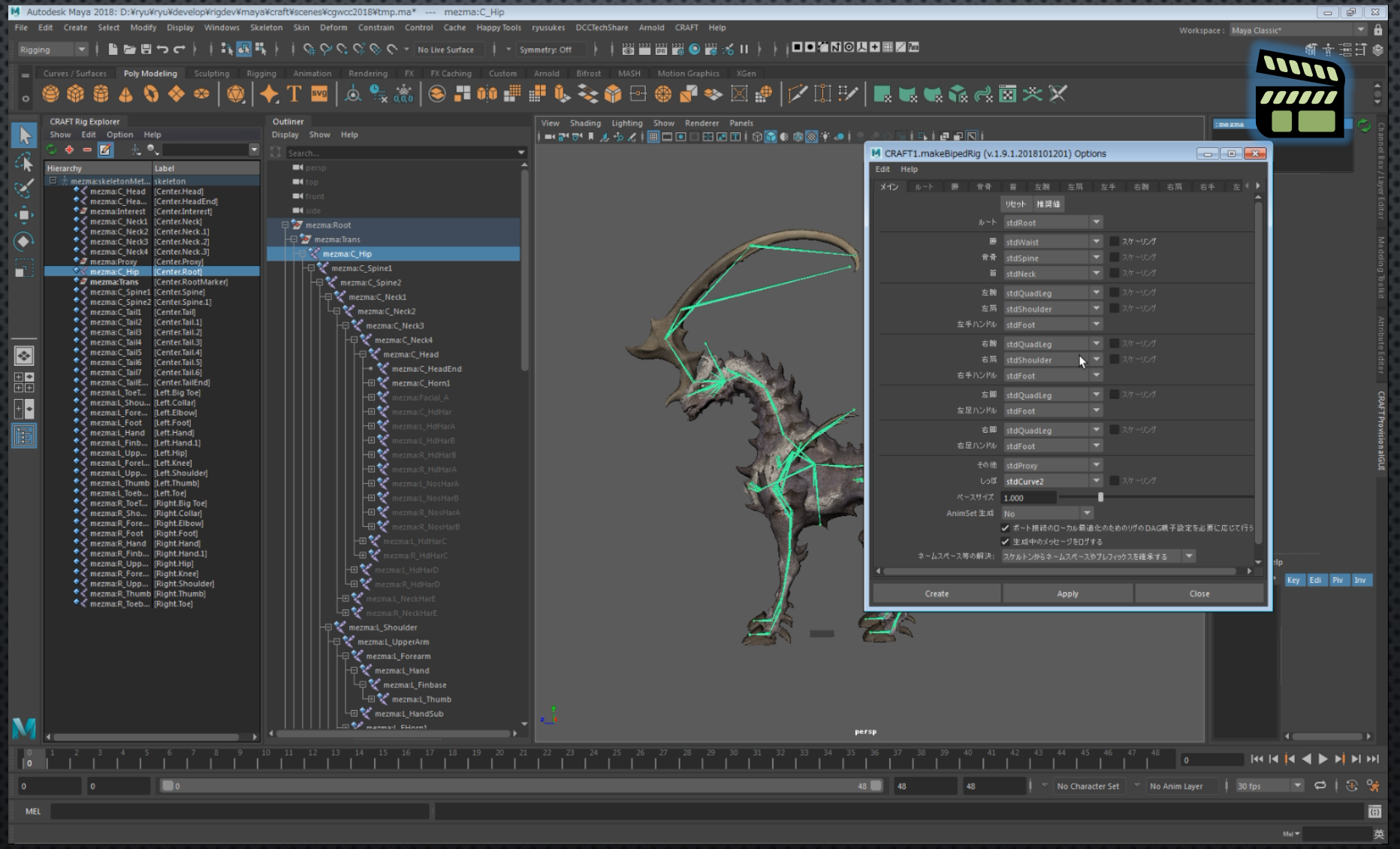


BIPED / QUADRUPED リグの簡易生成

- 胴体に四肢を持つキャラクタに特化した簡易ツール。
- 部位ごとに使用可能なモジュールを選択するだけで、リグ構築が行える。
- 足りない箇所は、後から手動で追加する。

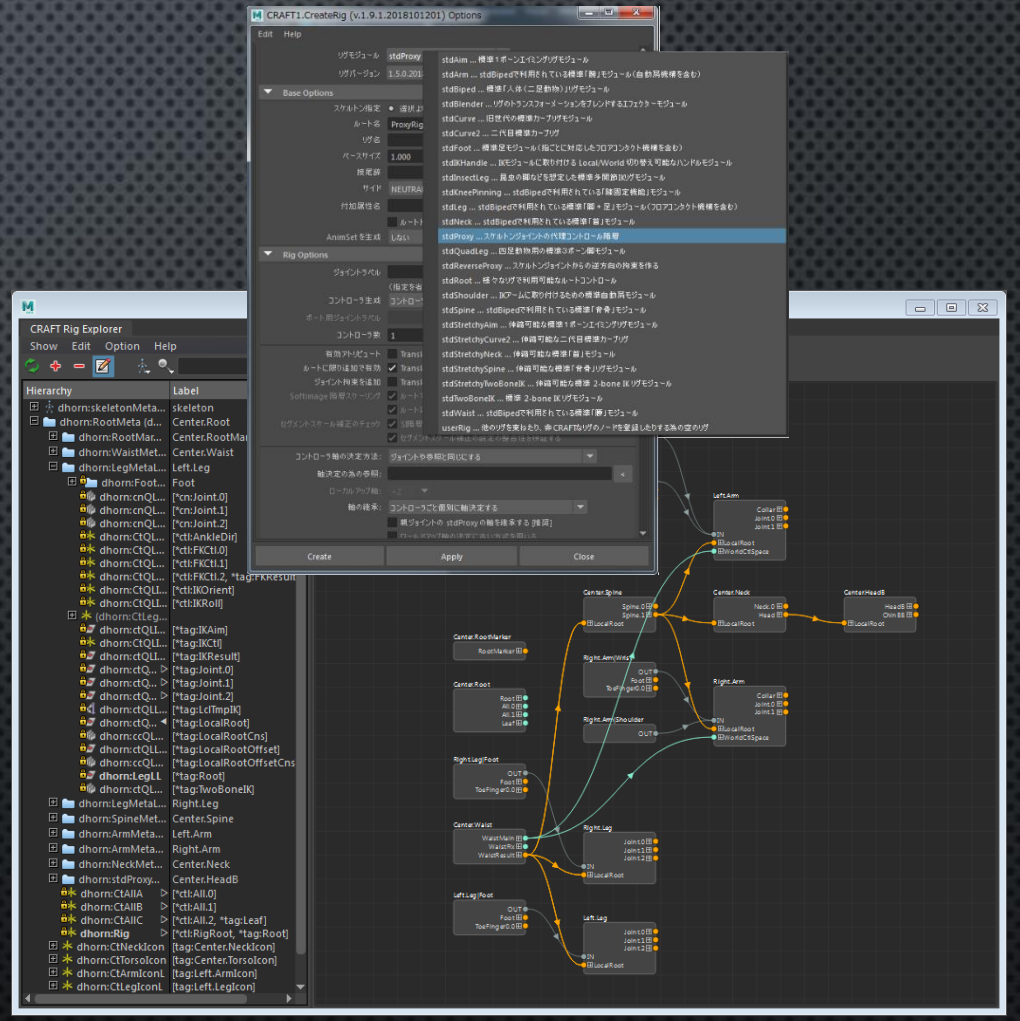


Biped / Quadruped リグの簡易生成

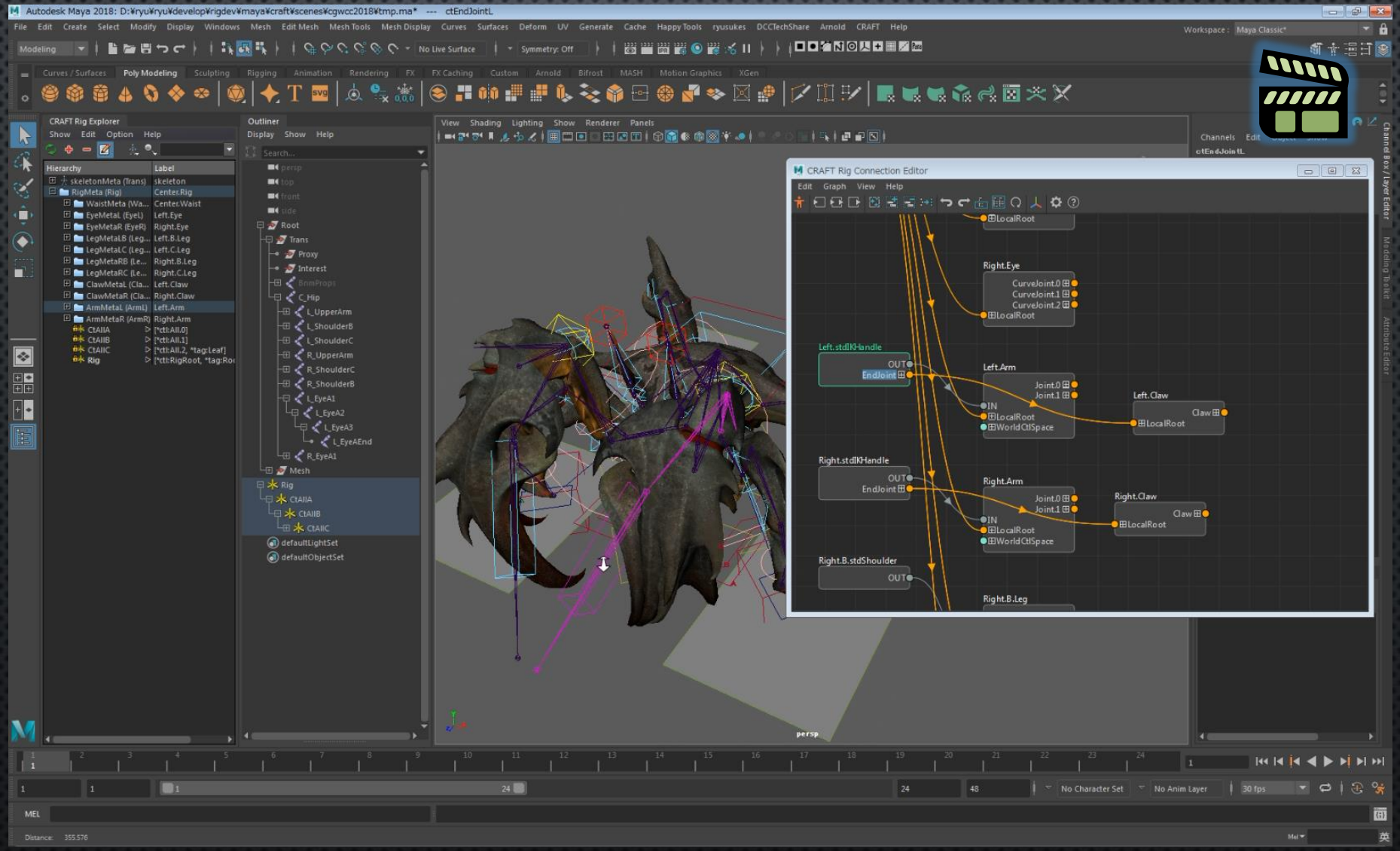


モジュールを組み合わせてリギング

- 各部位をモジュールで作っていく。
- モジュールの接続設定。
  - DAGノードの親子設定。
  - Rig Explorer でグルーピング。
  - Rig Connection Editor で接続。
  - 最後に最適化（モジュール間接続のローカルコンストレイン化）
- Auto-Connect 機能も有り。



モジュールを組み合わせてリギング



CRAFT リグモジュールの紹介

胴体系

- stdWaist

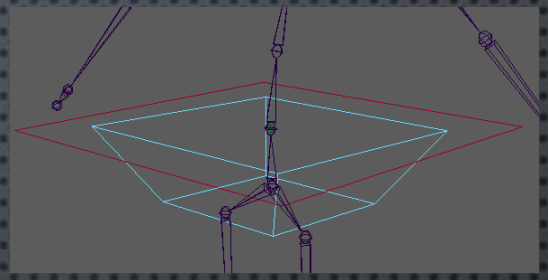
  胴体のルート（Hip や Waist）に。
  scaling も対応。

- stdSpine / stdStretchySpine

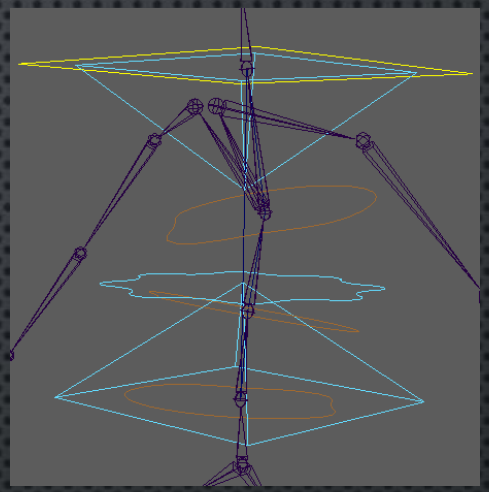
  背骨。ストレッチ対応で scaling も可。

- stdNeck / stdStretchyNeck

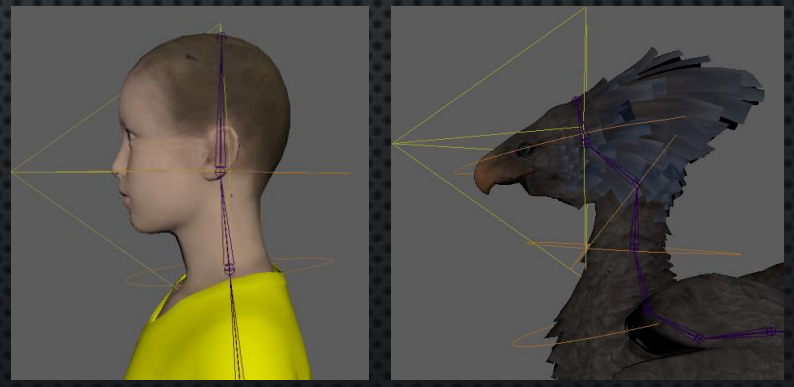
  首から頭。ストレッチ対応で scaling も可。



stdWaist



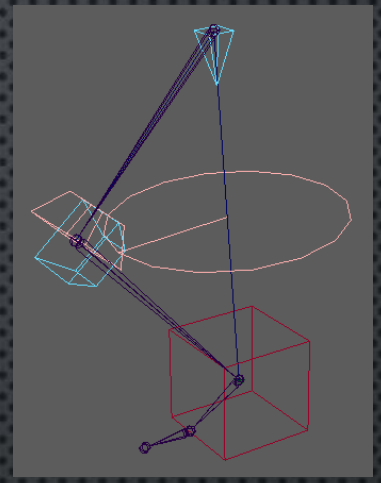
stdSpine



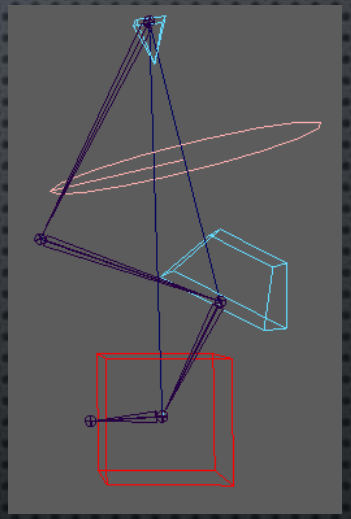
stdNeck

IK 系

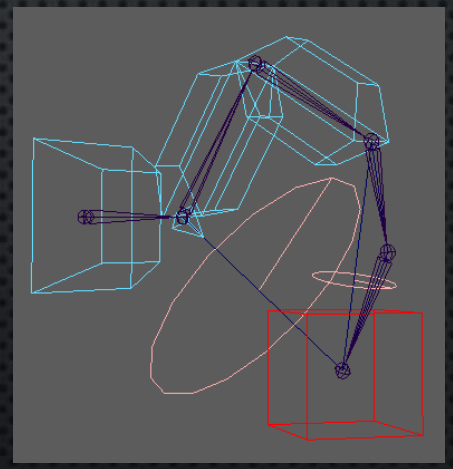
- stdTwoBoneIK / stdStretchyTwoBoneIK
  汎用 2-bone IK 。ストレッチ対応で scaling も可。
- stdQuadLeg
  四足動物の 3-bone IK 。
- stdInsectLeg
  節足動物の脚などに。関節数に制限無し。
- stdArm
  旧世代の腕。肩から手首まで含み汎用的でない。
- stdLeg
  旧世代の脚。Foot まで含み汎用的でない。



stdTwoBoneIK



stdQuadLeg



stdInsectLeg

IK 補助系

IK モジュールに取り付けるタイプ。

- stdIKHandle

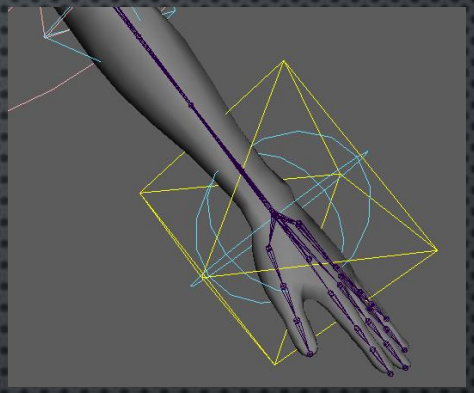
  先端に Local/World 切り替え機能を追加。

- stdFoot

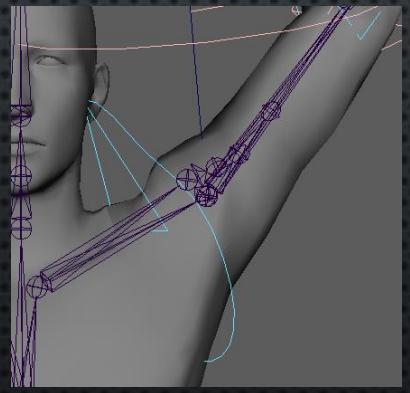
  先端に「足」を追加。
  フロアコンタクト機能。
  指の構造に柔軟に対応。

- stdShoulder

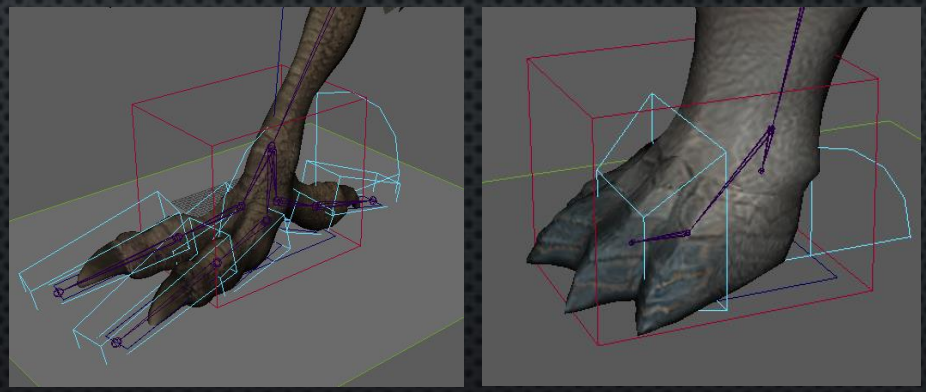
  自動肩機能を追加。



stdIKHandle



stdShoulder



stdFoot

プロキシ系

- stdProxy

  関節に１対１で対応する FK コントローラ。
  柔軟な回転軸決定機能。scaling サポートなど。

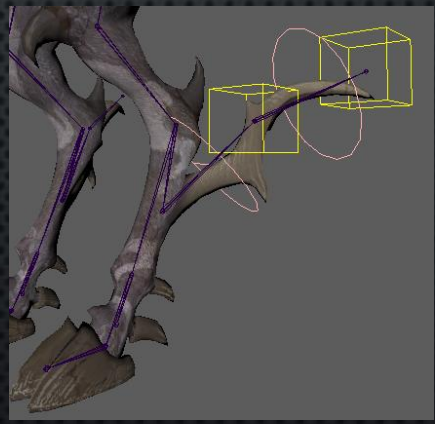
- stdReverseProxy

  骨からリグ階層への逆コンストレイン。
  補助骨の子に手動コントローラを追加したい場合に
  のみ用いる。

- stdAim / stdStretchyAim

  １関節のエイミングコントローラ。
  ストレッチ対応で scaling も可。

stdProxy



stdAim

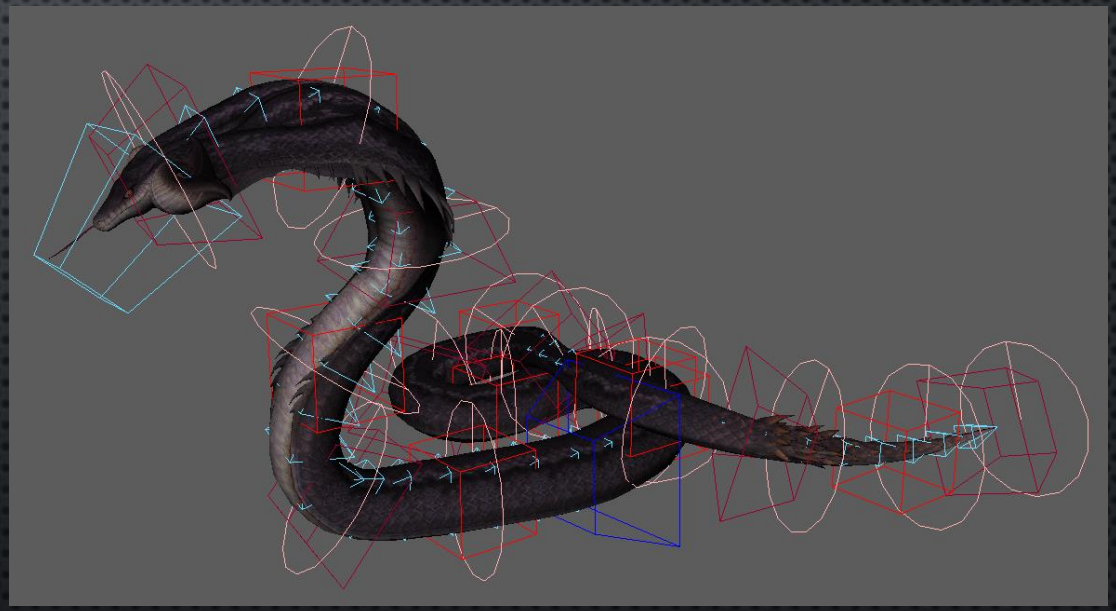
カーブ系

- stdCurve

  旧世代のカーブモジュール。

- stdCurve2 / stdStretchyCurve2

  旧世代より高機能で安定。
  ストレッチ対応で scaling も可。



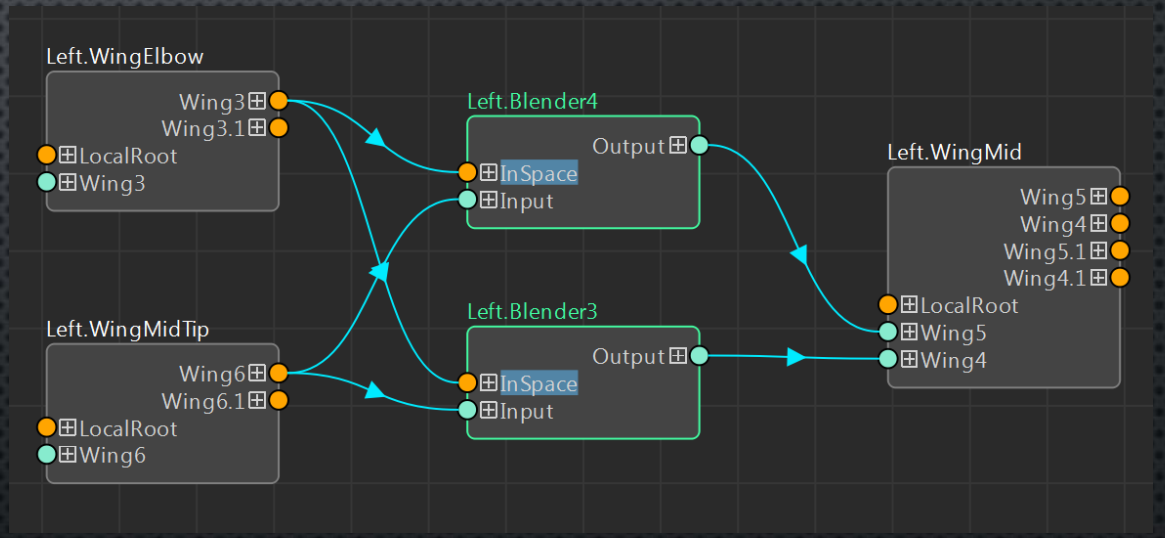
stdCurve2

エフェクタ系

モジュール間の接続による表現力を広げるためのもので、モジュールを新規に開発するほどでもないようなちょっとしたニーズに対応する。

- stdBlender

  多彩なブレンド機能を提供。
  これ自体はコントローラを持たない。



stdBlender

その他

- stdRoot

  リグ構造全体をまとめる。

- stdBiped

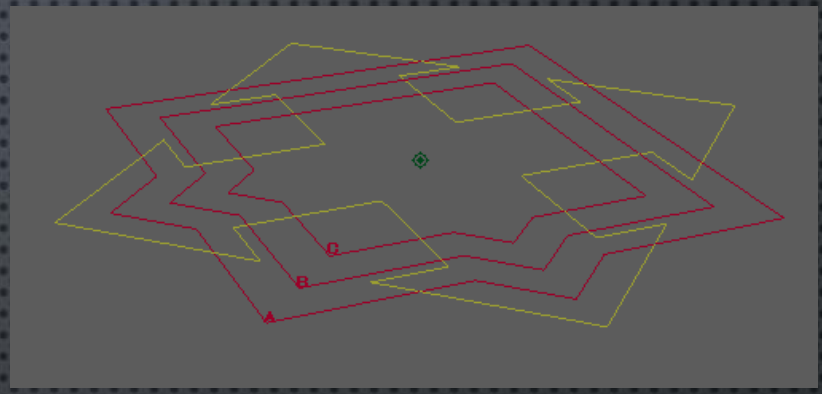
  モジュールを組み合わせた Biped 。
  stdArm や stdLeg を使っている古いタイプ。

- stdKneePinning

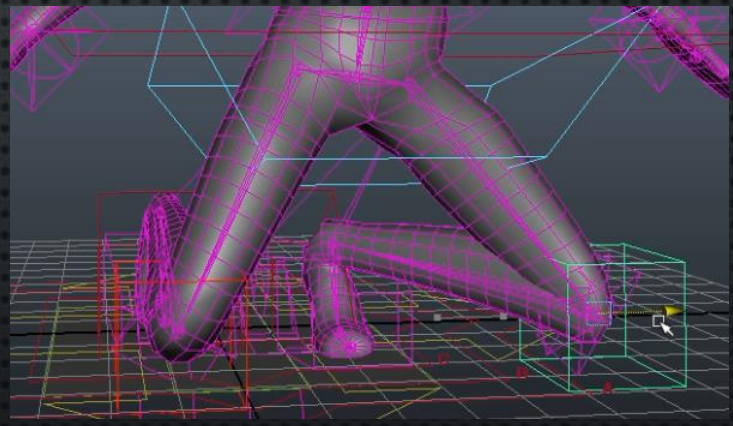
  膝固定機能。stdBiped で使用。

- userRig

  ユーザーが作成した任意のリグのコントローラを
  登録するためのもの。



stdRoot



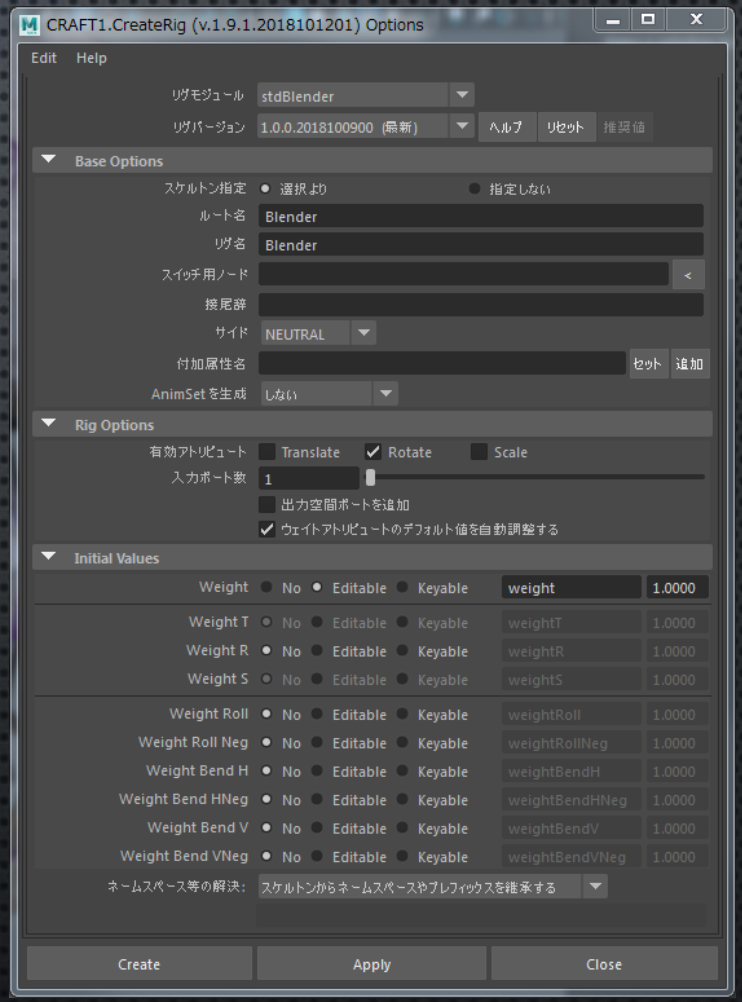
stdKneePinning

CRAFT リグモジュールの紹介

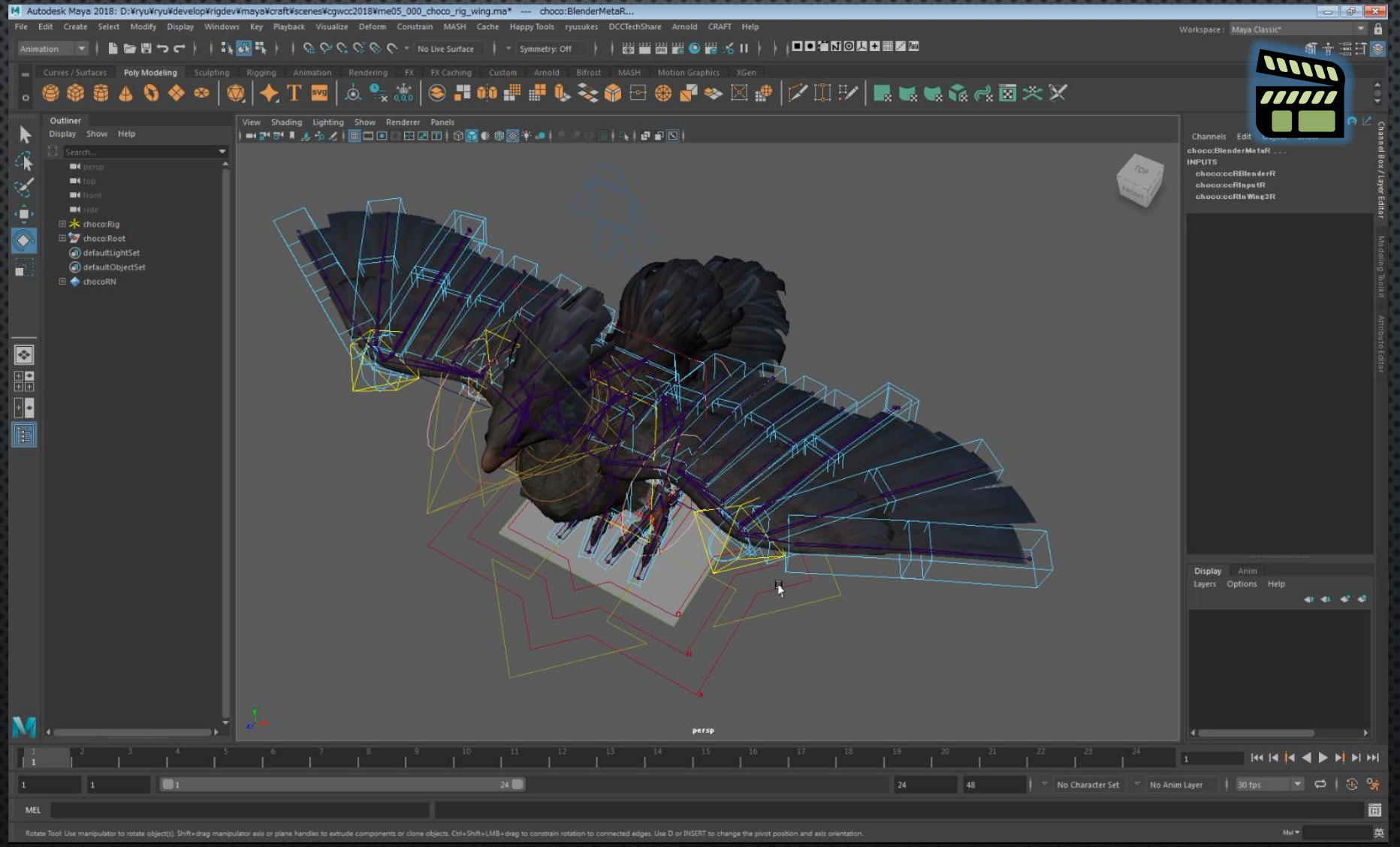
stdBlender について

stdBlender について

- Translate, Rotate, Scale の他、曲げ捻り分離の方向別など細かなブレンド制御。
- 入力空間下に１～ｎ個の入力。
  コンストレイン機能とは異なり、ウェイトは合計 1.0 に正規化されず、多彩な表現が可能。
- 入力空間とは別の出力空間を定義可能。
  例えば、シンメトリの拘束も作れる。



stdBlenderについて



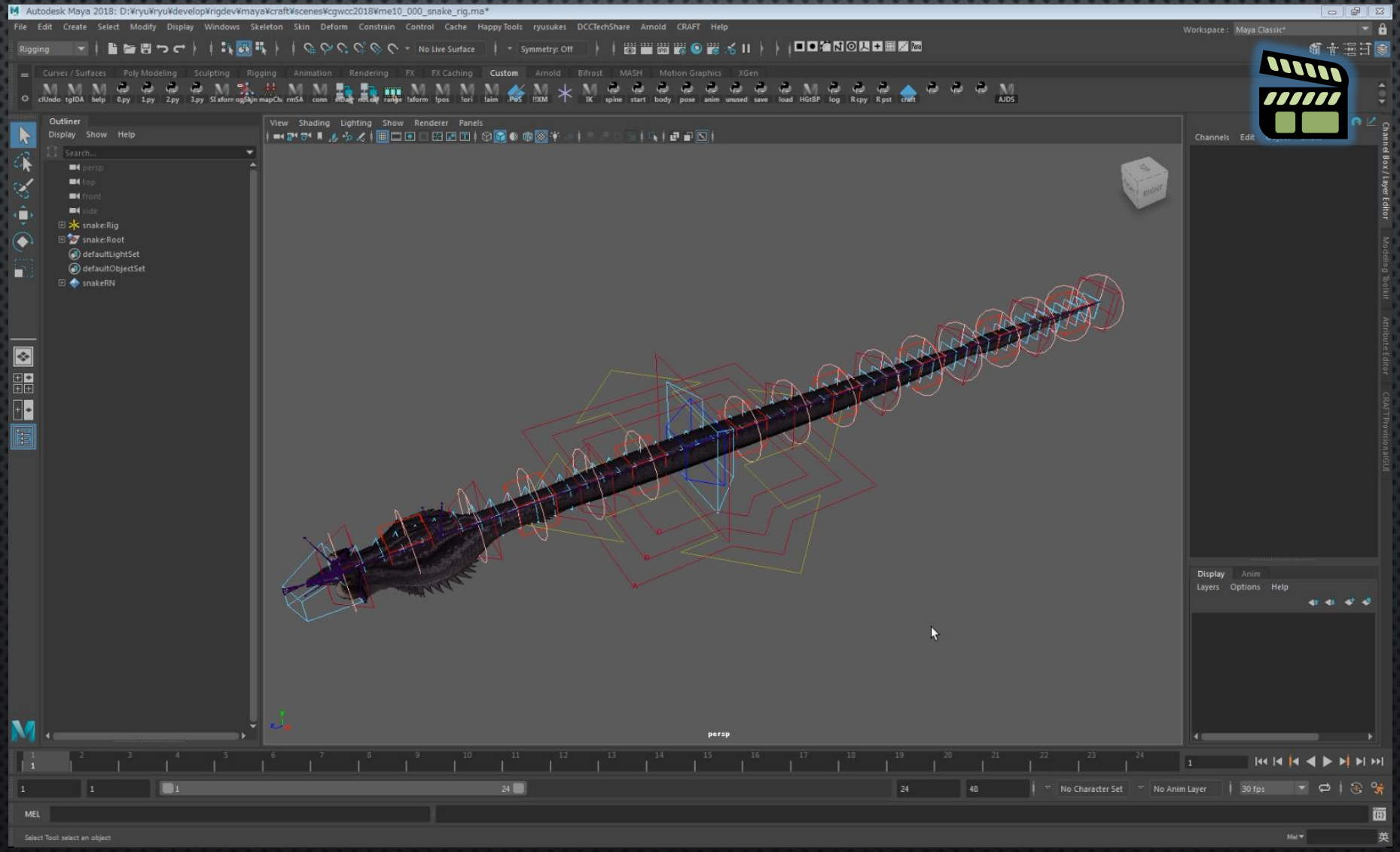
CRAFT リグモジュールの紹介

stdCurve2 / stdStretchyCurve2 について

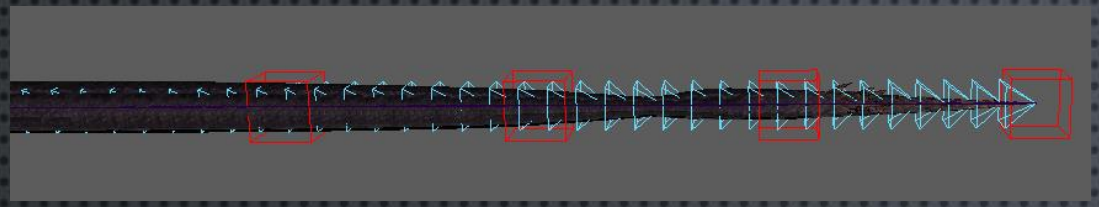
stdCurve2 について

- 多様な Twist コントローラ
  - 回転可能なポイントコントロール
  - Roll コントロール
  - アップベクトルコントロール
- Bezier カーブベース（カーブを通るコントローラ）
- 安定した局所的ストレッチ（均一な伸縮は今後の課題）
- リグ生成時に初期姿勢を維持

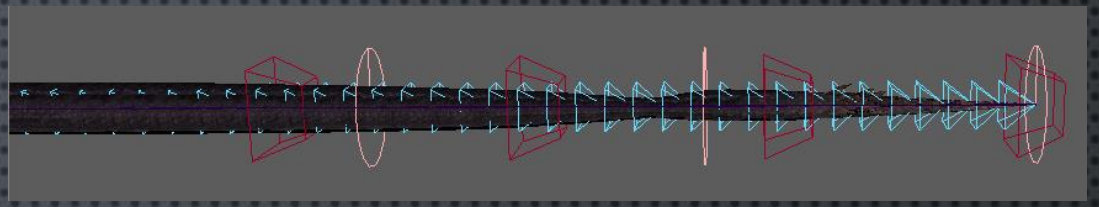
各コントローラの挙動



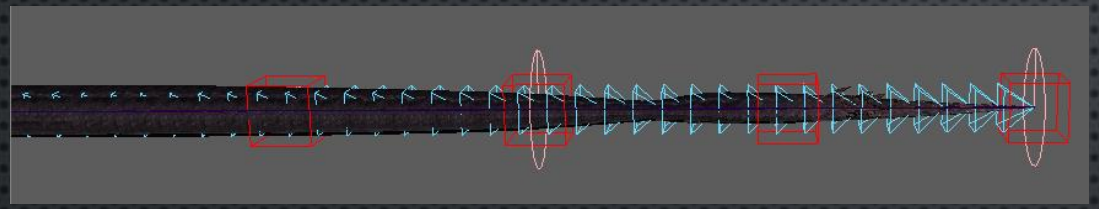
多様な Twist コントローラ



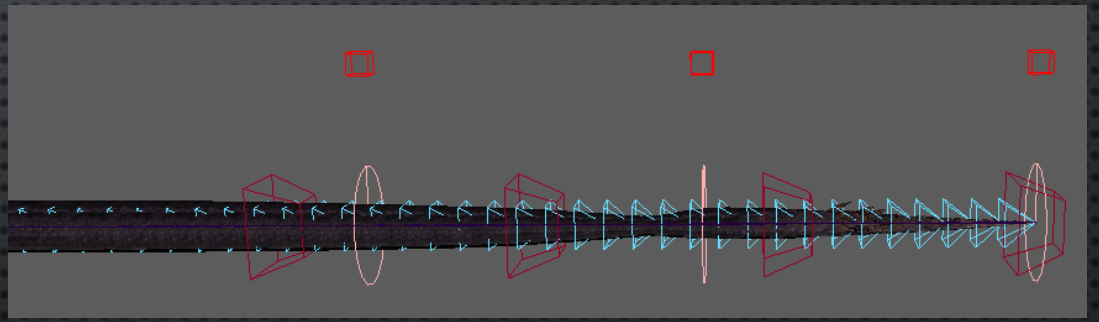
ひねり制御無し



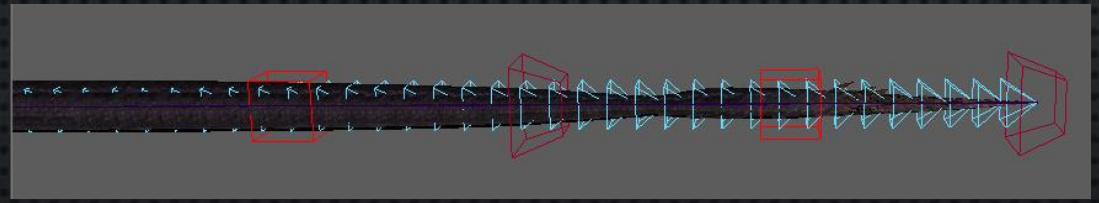
ポイントコントローラを全て回転可能に
且つ Roll コントローラを 3 個挿入



Roll コントローラを 2 個挿入



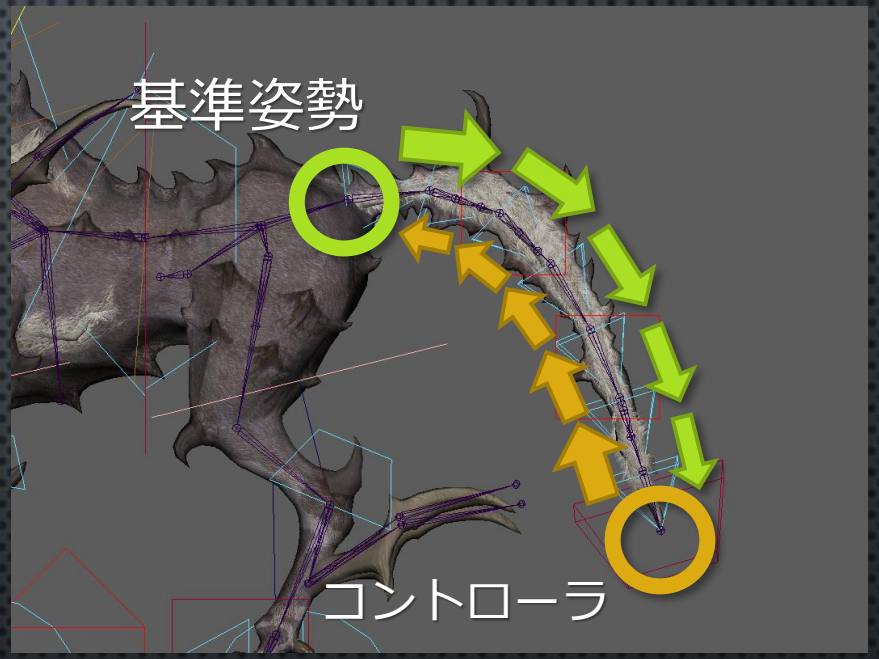
Roll コントローラに対して
アップポイントコントローラを有効化



ポイントコントローラ 2 個を回転可能に

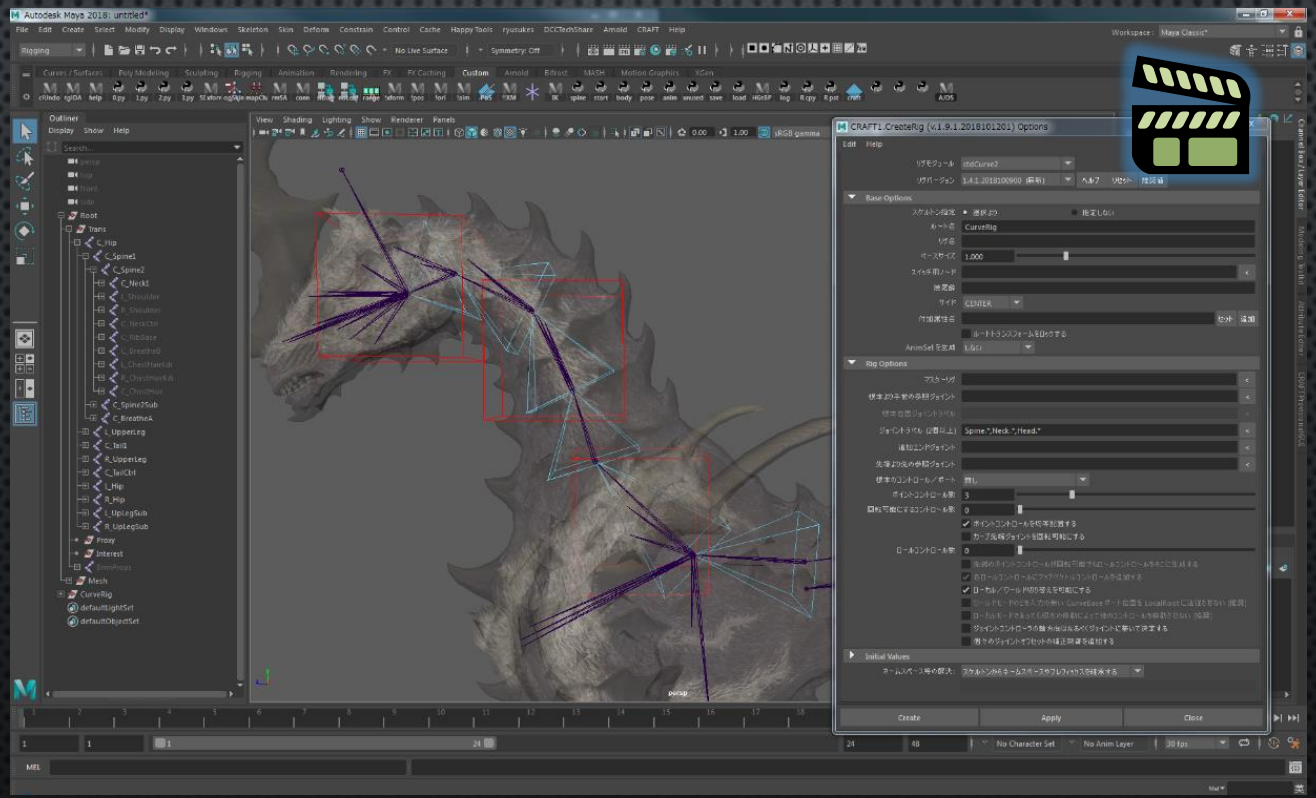
回転可能コントローラの意義

- アップベクトルコントローラを省きたかった。（好みによっては利用可能）
- 根本の姿勢を基準とし、先端に向けて徐々に曲げていくことで各関節のひねり姿勢を決定。
- しかし、動きが激しいと先端にいくほど不安定になる。そこで、先端を回転可能にして固定し、間は補間すれば安定する。
- さらに、指定数だけのコントローラも回転可能にすることで、長いカーブの中央も安定させる。



初期姿勢の維持について

- コントローラ入力用とスプラインIK用に Bezier カーブを二重化。
- カーブのバインドには独自 Curve Deformer を使用。

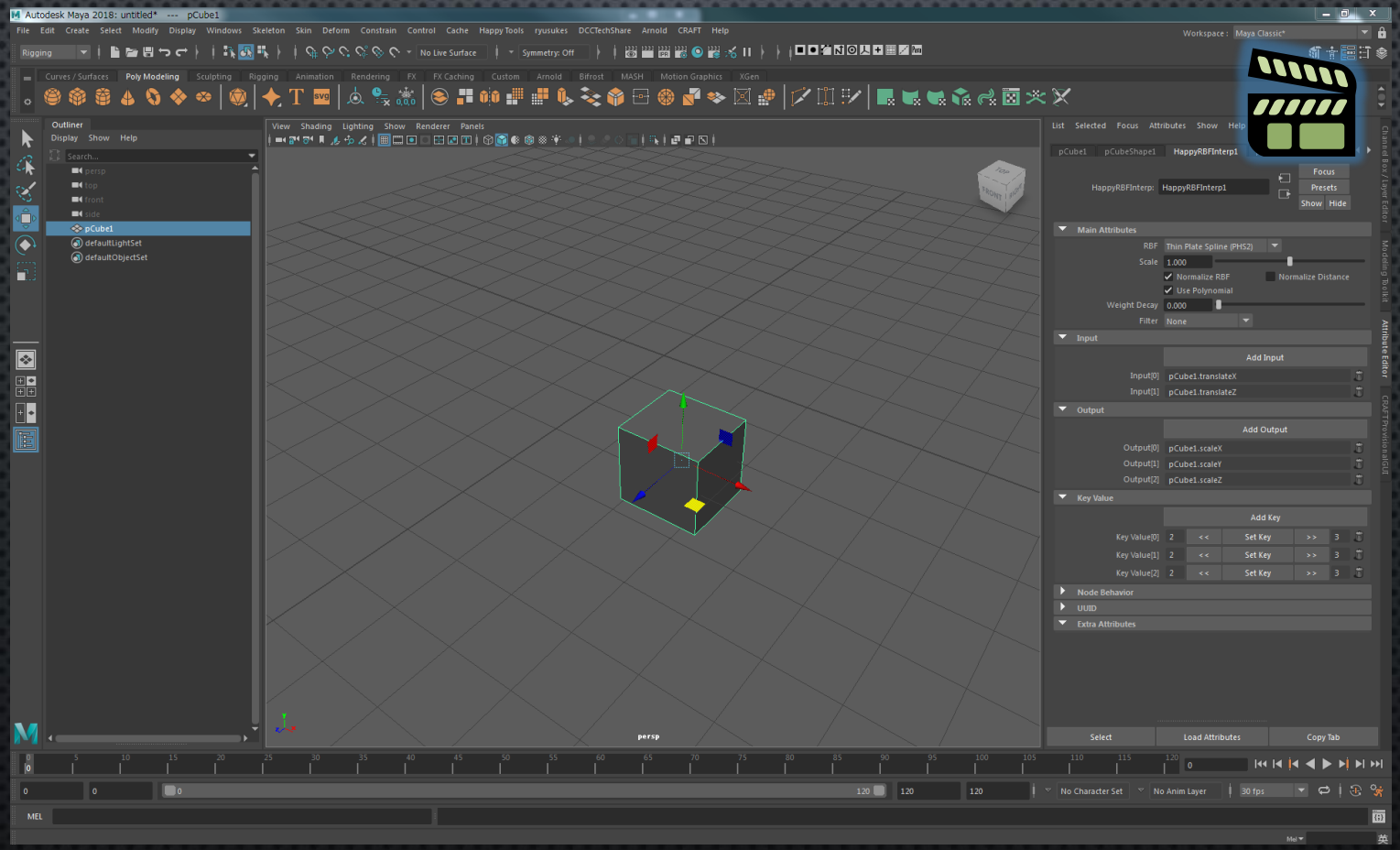


その他のツールの紹介

RBF 補間

RBF 補間ノード

n 個のキー入力で m 個の出力値を駆動する多次元ドリブンキー。



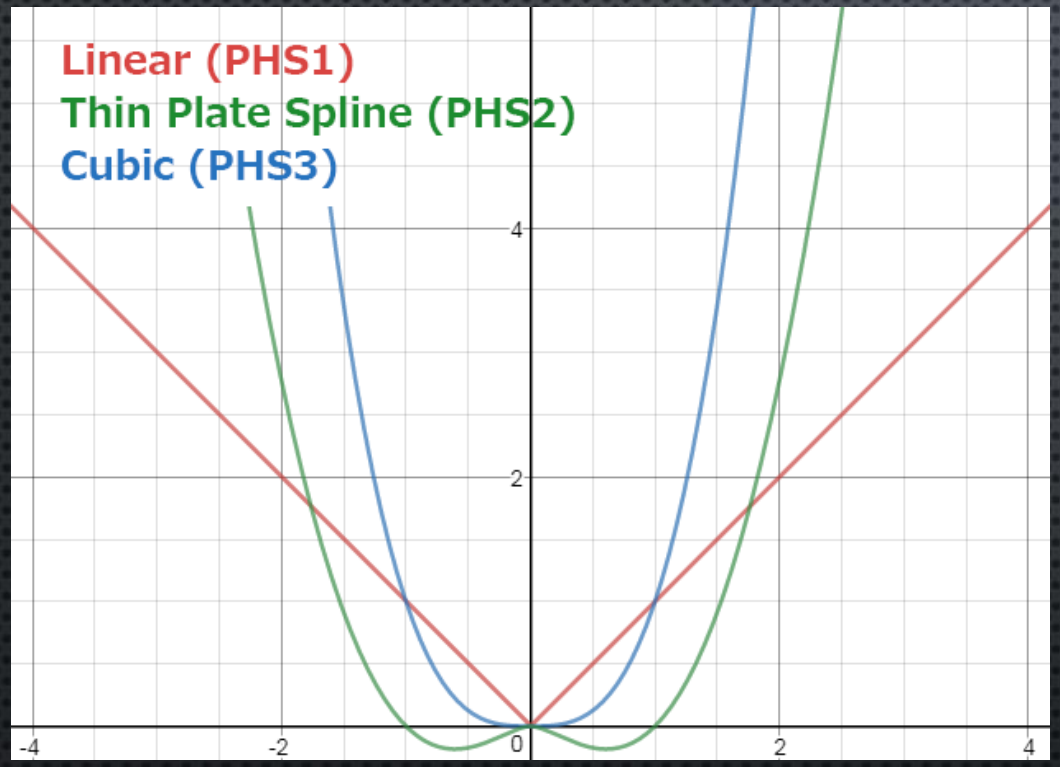
RBF 補間の特徴

- RBF補間とは、RBF (Radial Basis Function) と呼ばれる単純な関数を用いた関数近似手法。
- N 個あるキー c_i の RBF φ の総和によって、それらの点を通る関数 $f(x)$ を得る。

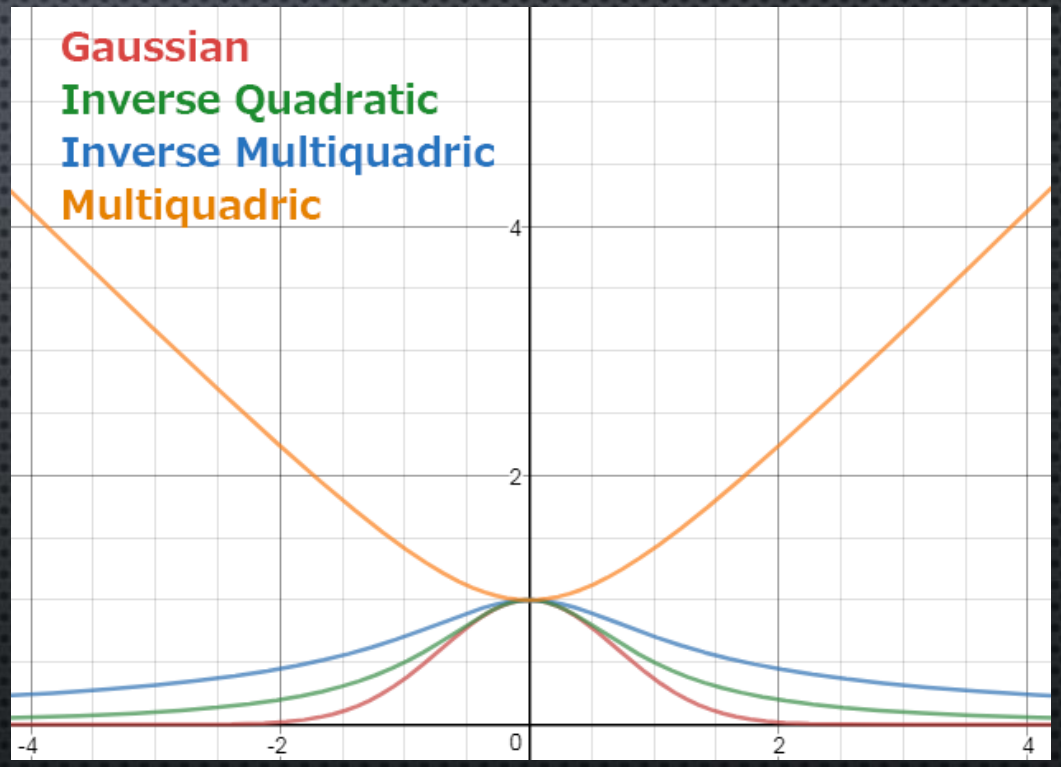
$$f(x) = \sum_{i=1}^{N} w_i \varphi(\|x - c_i\|)$$

- 通常、与えたキーを通る滑らかな補間が得られる。
- 補間具合の細かい調整は出来ず、RBF の種類の使い分けで調整する。
- 通常、外挿 (Extrapolation) は得られない。

RBF の使い分け



Poly-harmonic Spline
（重調和スプライン）

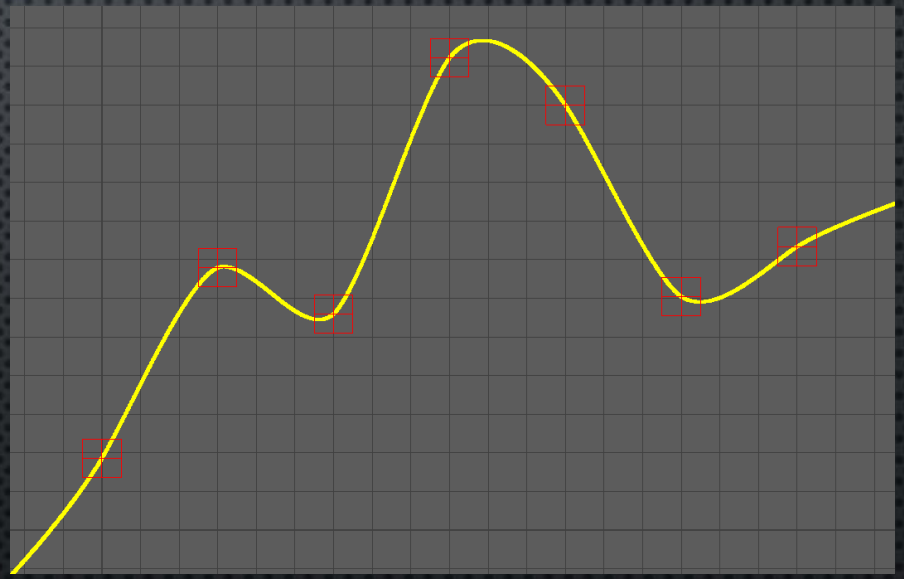


重調和スプライン以外
（上向きの Multiquadric は他と性質が異なる）

RBF の使い分け ～ 重調和スプライン

キーを自然に結んだ曲線が得られる。実寸法の影響を受けない。
通常のドリブンキーのように「いい感じ」に補間したいならこちら。

Linear
（１次の重調和スプライン）

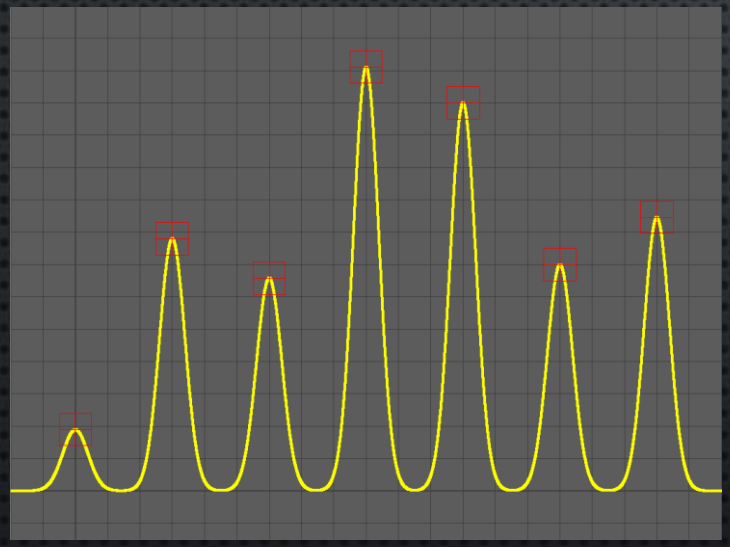


Thin Plate Spline
（２次の重調和スプライン）

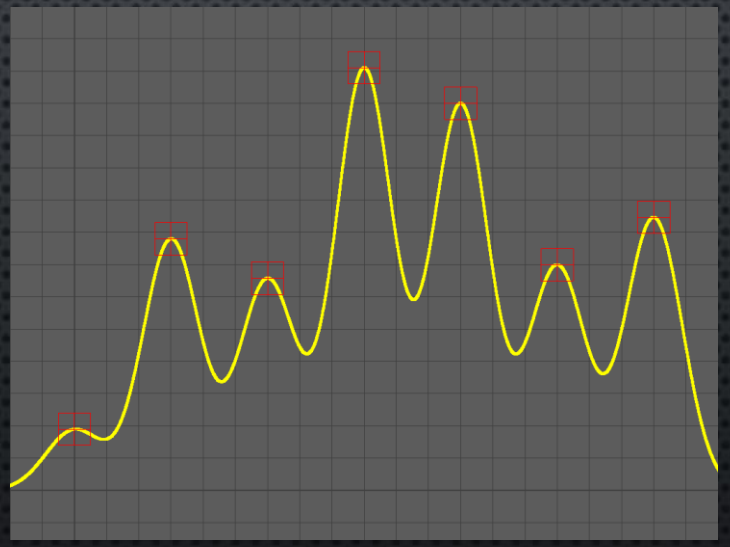
RBF の使い分け ～ ガウシアン等

キー間がゼロになる曲線を得られる。実寸法が影響するので扱いにくい。
RBF半径（スケール）を調整出来る。正規化してゼロにならないようにも出来る。

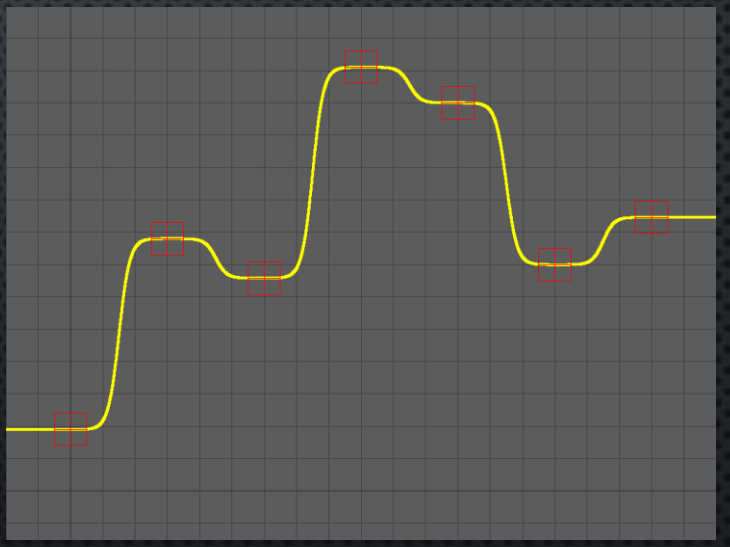
Pose Space Deformation（スキニング＋補正ブレンドシェイプ）のようにキーポーズ間をニュートラル（補正無し）とするような補間に向く。



Not Adjusted



Radius (scale) Changed

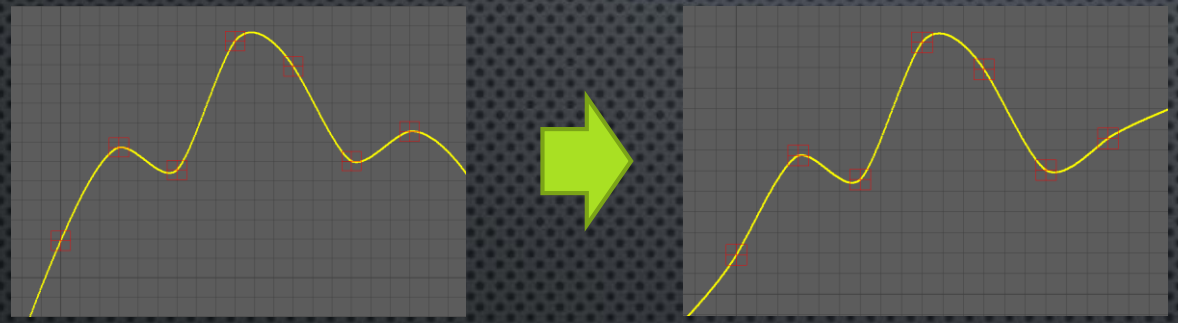


Normalized

RBF 補間の追加機能

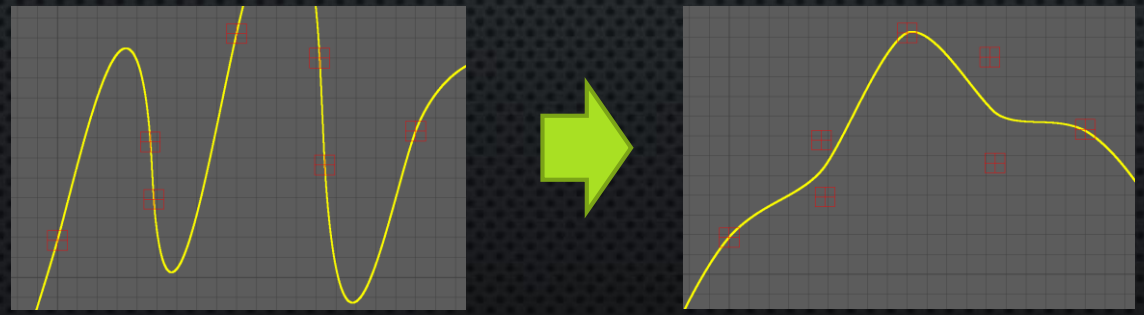
- Use Polynomial

  線形補間を併用することで、全体的な補間と外挿を少し改善する。

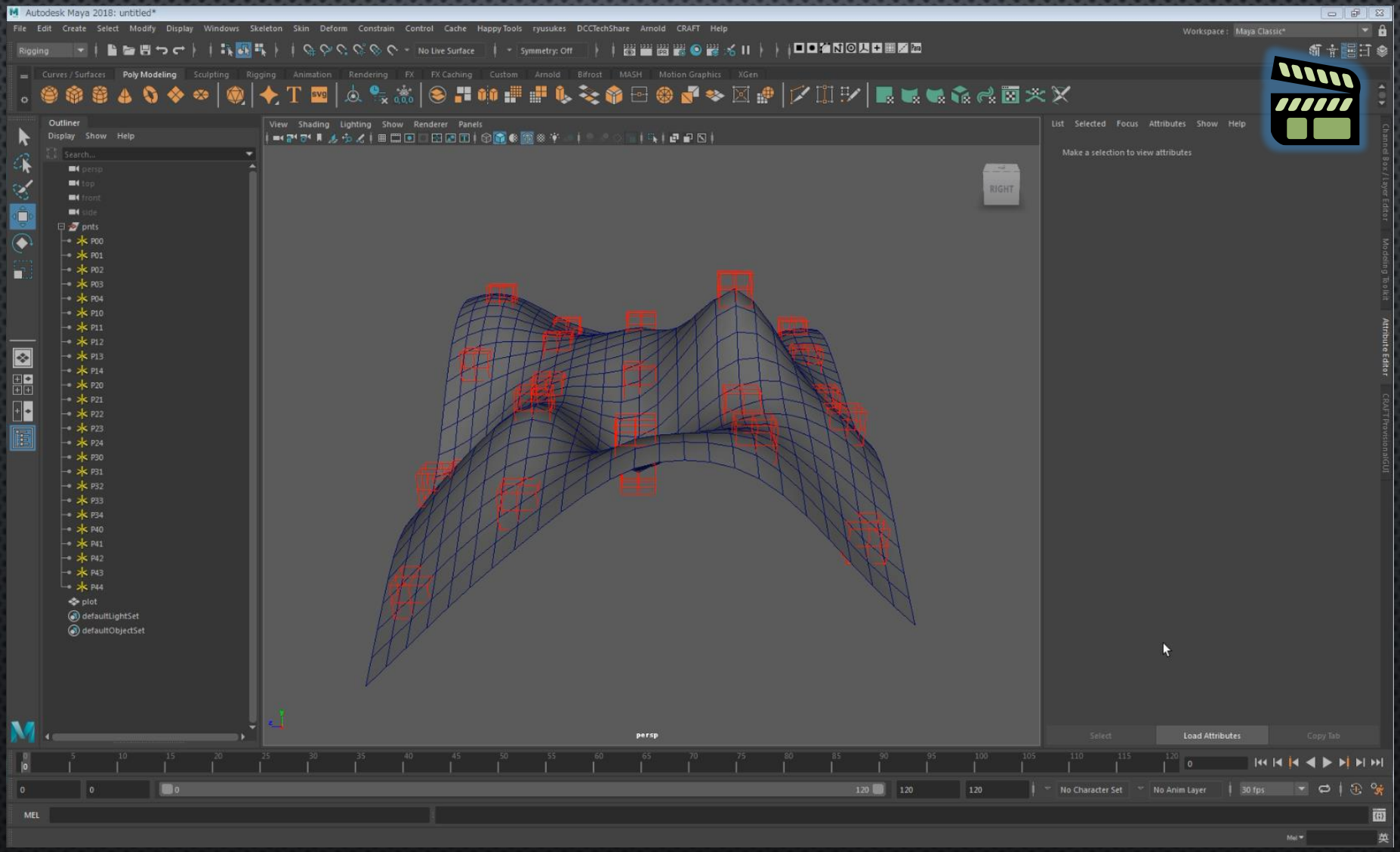


- Weight Decay

  正則化。オーバーフィッティング（過学習）を抑えて補間を滑らかにする。



RBF 補間の追加機能



その他のツールの紹介

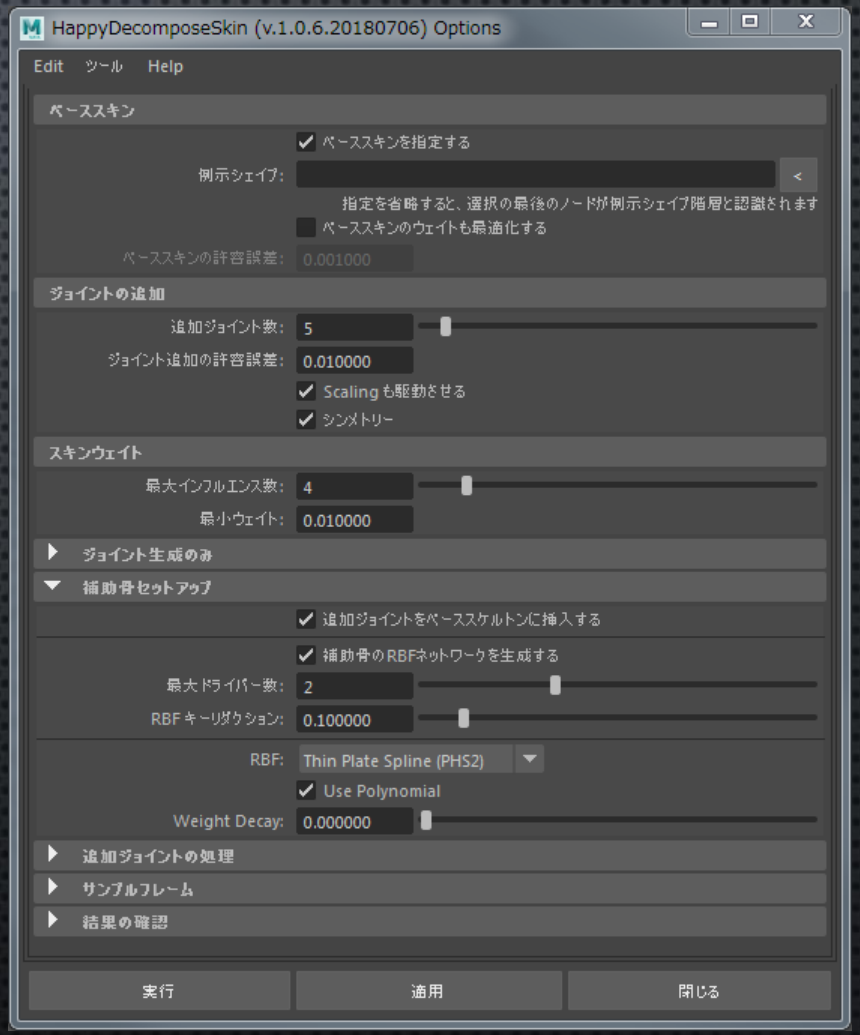
スキニング分解

スキニング分解ツールの機能

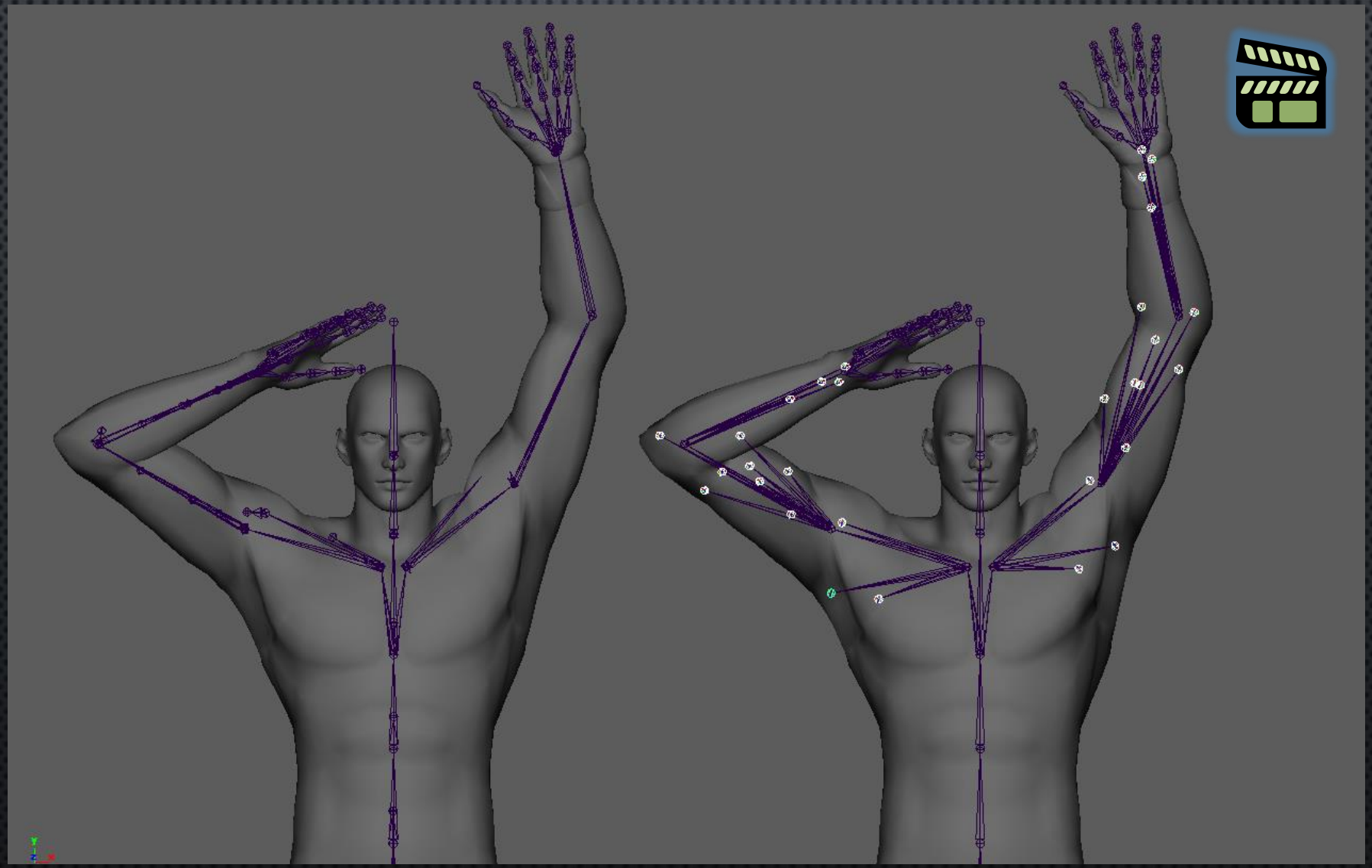
- 任意のデフォーメーションをスキンアニメーションに変換
- 既存スキンのウェイトの最適化
- RBF 補間による補助骨の自動生成

  スケルトン駆動の高度なデフォーメーション（筋肉、PSD等）を学習して近似する。
- フェイシャルリグ等のボーン化

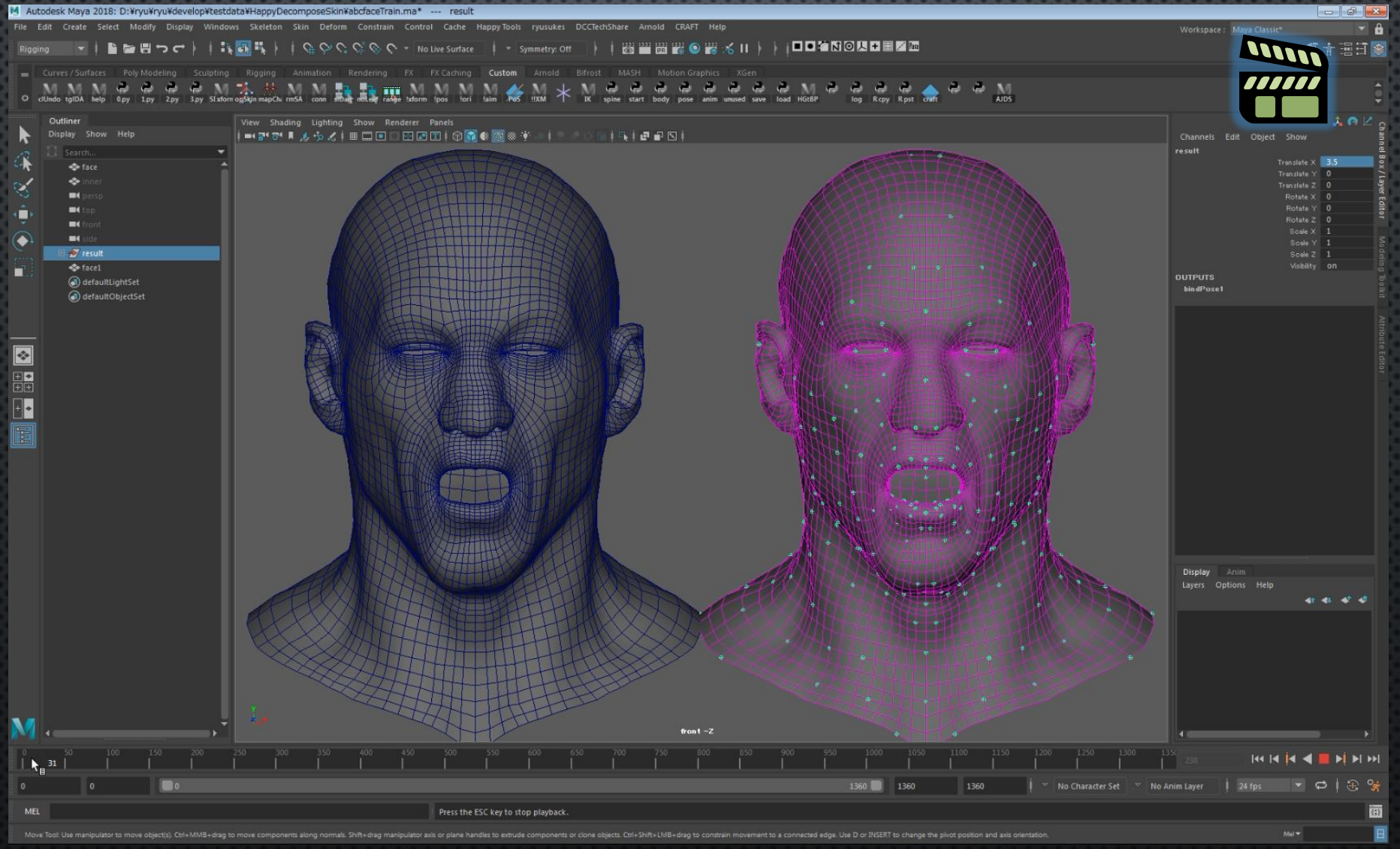
  骨の自動生成とウェイト調整。



RBF で駆動する補助骨の自動生成



フェイシャルリグ等のボーン化



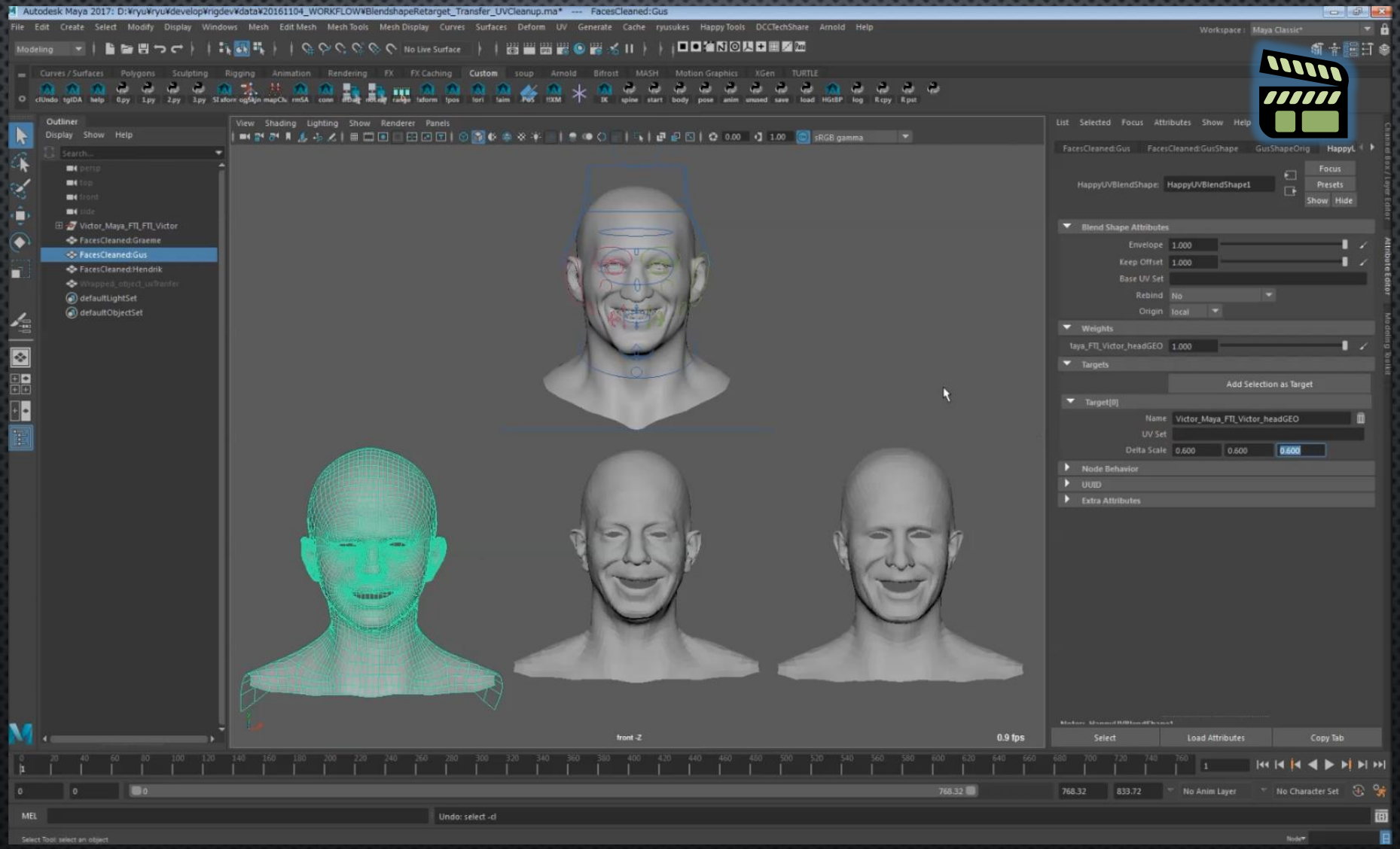
その他のツールの紹介

UVブレンドシェイプ

UVブレンドシェイプの機能

- 頂点数やトポロジが異なるが UV レイアウトが一致するモデルをブレンドシェイプする。
- オフセット機能
  ベースメッシュとターゲットメッシュを初期状態で一致させ、そこからの変形差分のみをミックスする機能。異なる顔の表情変化のみを抽出出来る。
- ウェイトペイント機能。

UVブレンドシェイプの機能



その他のツールの紹介

ノードベース GUI フレームワーク

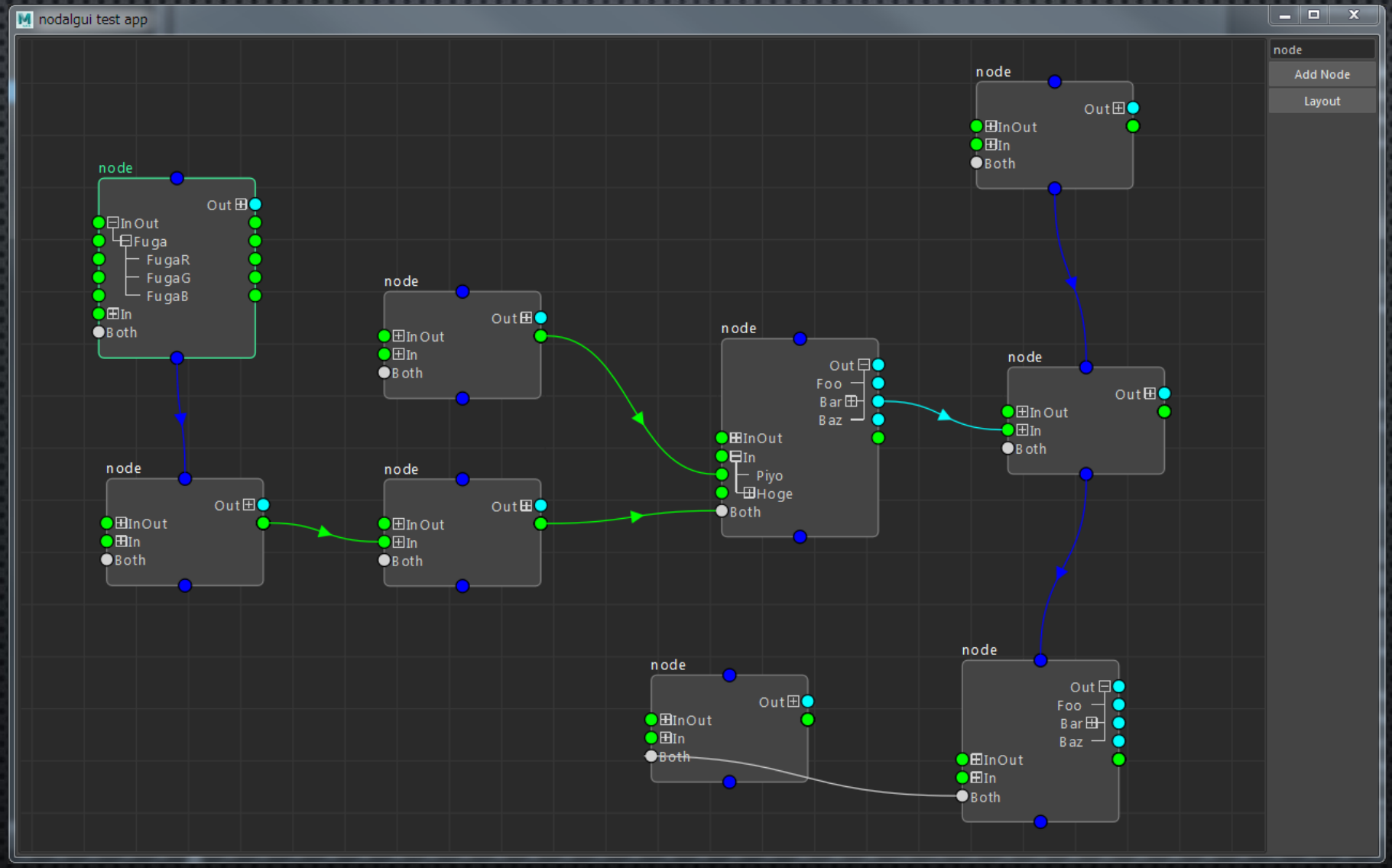
nodalgui フレームワーク

- Python でノードベース GUI を作るためのフレームワーク。
- CRAFT の Rig Connection Editor はこれで作られた。
- PyQt4, PyQt5, PySide, PySide2 のどれでも動く。
- Maya® に依存しない。

nodalguiテストアプリケーション

```
from Happy.Qt.nodalgui import Scene, View, Node, Port
from Happy.Qt.nodalgui.common import QtWidgets, QVector2D, Qt
from Happy.Qt.util import showAppWindow

class Window(QtWidgets.QWidget):
    def __init__(self, *args, **kwargs):
        super(Window, self).__init__(*args, **kwargs)

        self.setWindowTitle('nodalgui test app')
        self.resize(800, 600)

        hbox = QtWidgets.QHBoxLayout()
        hbox.setSpacing(2)
        hbox.setContentsMargins(2, 2, 2, 2)
        self.setLayout(hbox)

        view = View()
        self.view = view
        view.setScene(Scene())

        vbox = QtWidgets.QVBoxLayout()

        hbox.addWidget(view)
        hbox.addLayout(vbox, 10)

        self.nameFld = QtWidgets.QLineEdit('node')
        addBtn = QtWidgets.QPushButton('Add Node')
        addBtn.clicked.connect(self.addNode)

        layoutBtn = QtWidgets.QPushButton('Layout')
        layoutBtn.clicked.connect(lambda: view.autoLayout())

        vbox.addStrut(100)
        vbox.addWidget(self.nameFld)
        vbox.addWidget(addBtn)
        vbox.addWidget(layoutBtn)
        vbox.addStretch(1)

    def addNode(self):
        node = Node(self.nameFld.text())
        rect = node.rect()

        Port(node, direction=-1, vec=QVector2D(0., rect.top()), brush=Qt.blue)
        Port(node, direction=1, vec=QVector2D(0., rect.bottom()), brush=Qt.blue)

        top = node.addPort(1, name='Out', brush=Qt.cyan)
        Port(top, 'Foo')
        p = Port(top, 'Bar')
        Port(p, 'BarX')
        Port(p, 'BarY')
        Port(p, 'BarZ')
        Port(top, 'Baz')

        top = node.addPort(0, name='InOut', brush=Qt.green)
        p = Port(top, 'Fuga')
        Port(p, 'FugaR')
        Port(p, 'FugaG')
        Port(p, 'FugaB')

        top = node.addPort(-1, name='In', brush=Qt.green)
        Port(top, 'Piyo')
        p = Port(top, 'Hoge')
        Port(p, 'HogeX')
        Port(p, 'HogeY')
        Port(p, 'HogeZ')

        node.addPort(-1, name='Both', direction=0)

        self.view.addItemOnCenter(node)

showAppWindow(Window)
```

nodalgui テストアプリケーション



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, 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
- 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 (25 June 2017, 21:04 UTC). In Wikipedia: The Free Encyclopedia. Retrieved from https://en.wikipedia.org/wiki/Polyharmonic_spline

ご清聴ありがとうございました。

佐々木 隆典

ryusukes@square-enix.com

SQUARE ENIX ADVANCED TECHNOLOGY DIVISION