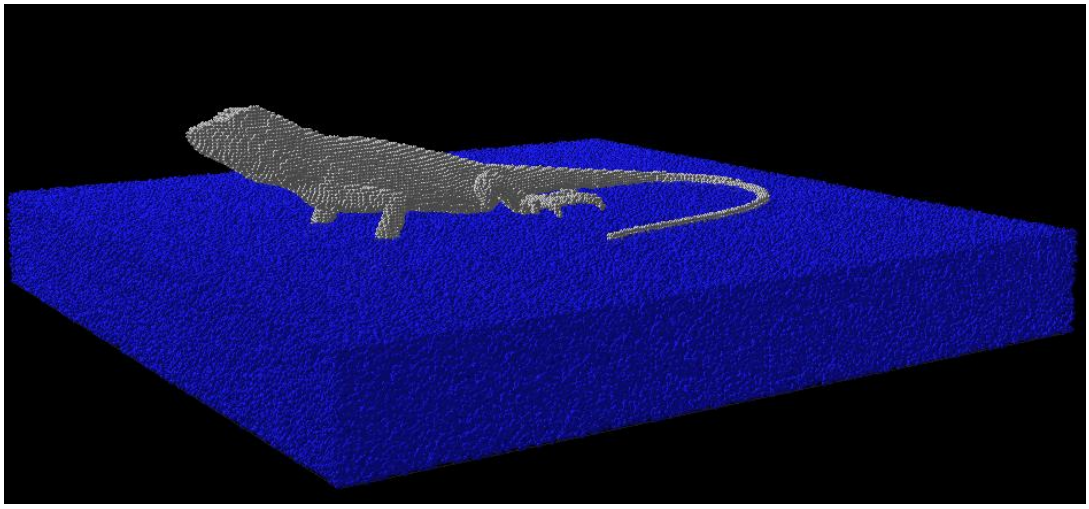
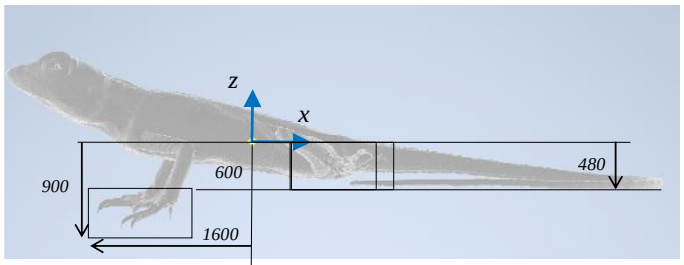
On-water run by MLS-MPM in Taichi

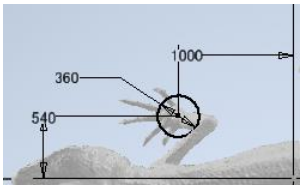
11 Mar. 2026

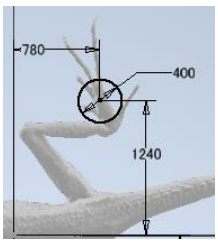
Taichi の MLS-MPM を使って疑似的 FSI（足先だけ運動学，その他動力学）で水面を 4 脚で走るトカゲをシミュレートしてみる．概略は，トビウオの双方向シミュレーション参照．

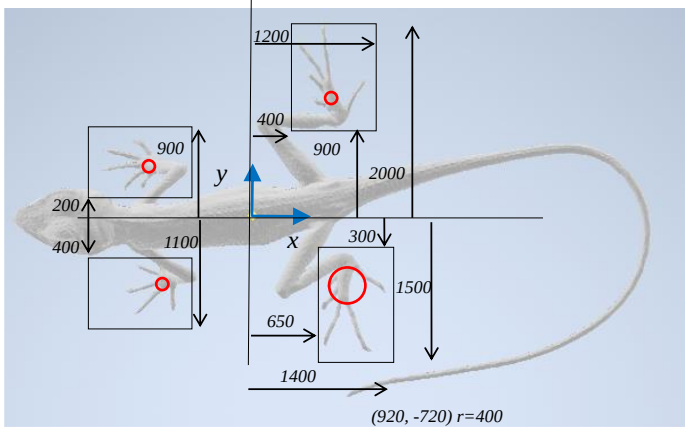
計算空間：0.8m*0.8m*0.8m
格子解像度：128*128*128→格子サイズ 0.8/128=6.25 [mm]
粒子サイズ：1 格子に 2^3=8 つ
水領域：0.8m*0.8m*0.1m
トカゲ：弾性体粒子，体長 0.5m ぐらい，周波数 10Hz
走行：足先だけ位相を 90deg づつずらした楕円軌道．姿勢はそのまま．体はそれに追従する．脚の運動は運動学で一方向，つながっている体は弾性体で双方向連成．

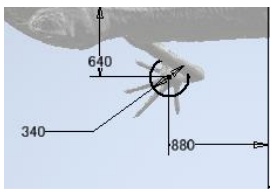


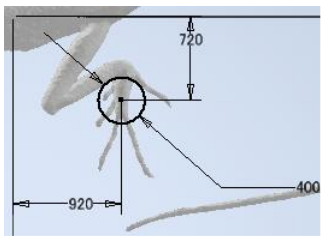


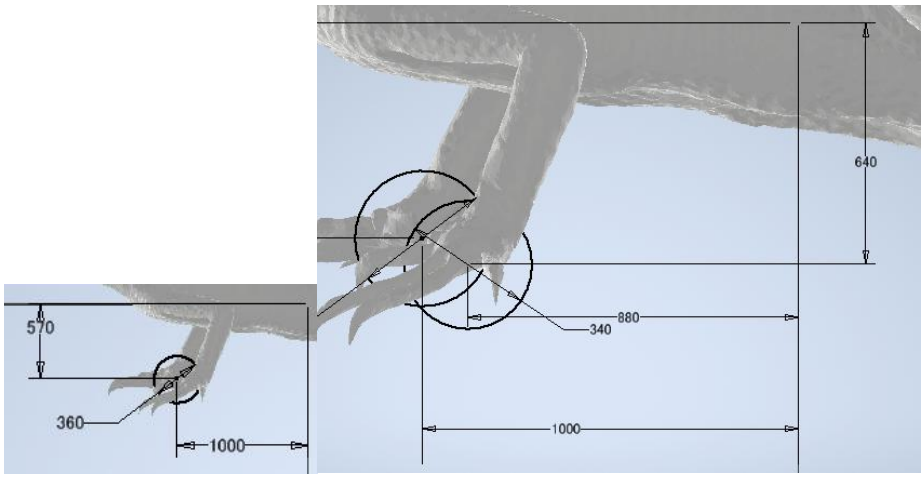


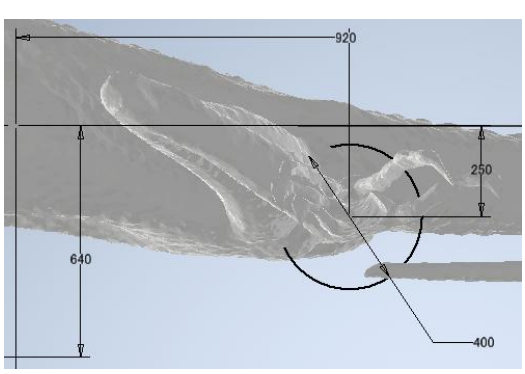


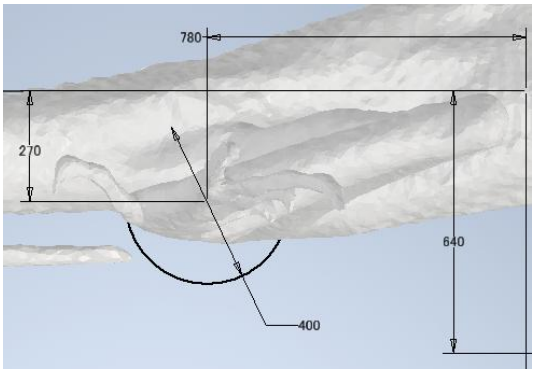












↑この単位は 10 倍になっている．

(1) 標本データ

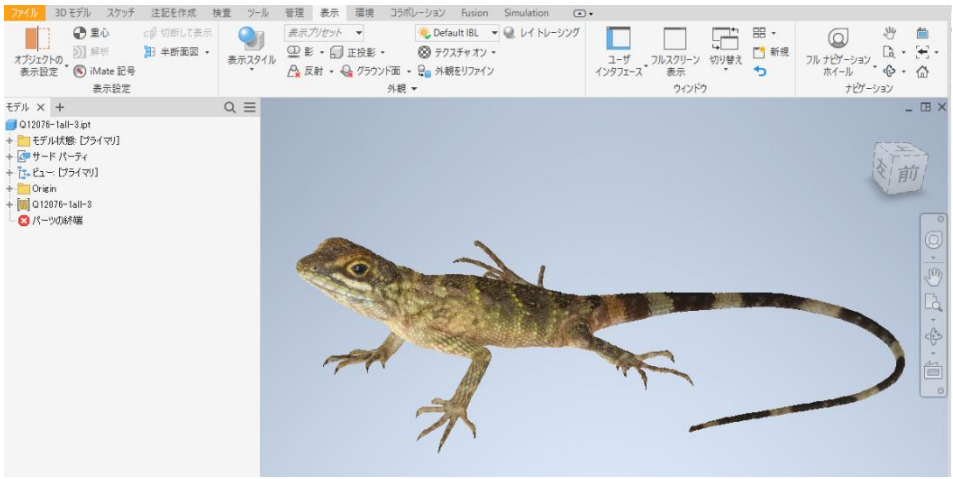
標本データを up している以下のサイトから，lizard で検索して適当なトカゲをダウンロード．

https://sketchfab.com/ffishAsia-and-floraZia/models

これ．

https://sketchfab.com/3d-models/cc0-sakishima-tree-lizard-15f772d7febd45a48d3cecf1cef6fe10

ここでは，テクスチャ付きの obj ファイルをつかう．inventor で読み込むとこんな感じ．

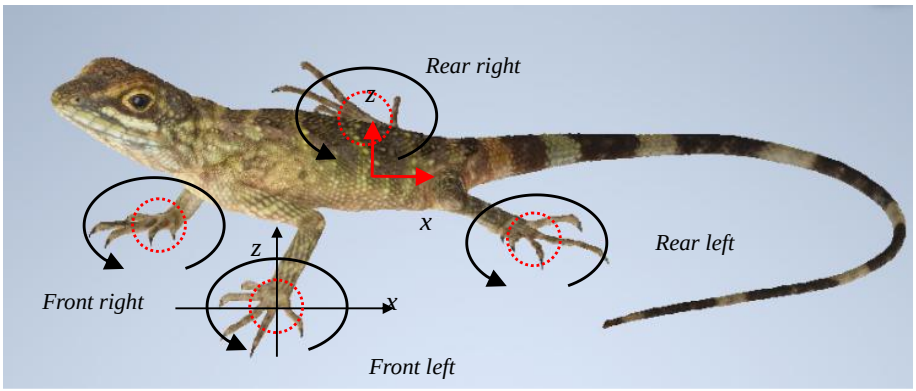


．必要ないボックスは消しておく．

．座標は y が上になっているので，z に回転しておく．

．サイズも適当なので，スケーリングしておく．これのデフォルトは 5m になっている．生物系の人は CAD 上でのサイズを気にしないらしい．
．obj ファイルを使うが，テクスチャを張り付けるための uv データが入っている．最後の可視化で利用する．

(2) 4 脚走行
4 脚それぞれの位相を 90 づつずらした楕円軌道にする．
$x_i = a\cos(2\pi f t + \varphi_i)$
$z_i = b\sin(2\pi f t + \varphi_i)$
ここで，f は周波数，a, b は楕円の長軸短軸，ϕ i は位相で 0, 180deg, 90deg, 270deg.
i=0: 左前脚，1:右前脚，2:左後ろ足，3: 右後ろ脚.
速度は，位置の微分.
$vx_i = -2\pi f a\sin(2\pi f t + \varphi_i)$
$vz_i = 2\pi f b\cos(2\pi f t + \varphi_i)$
赤い点線内の粒子に上記の運動を与える．他の粒子はその動きに弾性連続体として追従する．dt が粗いと足が切れる．柔らかすぎても（ヤング率 10^6 になっている．金属の 1000 倍柔らかい）重力に負けて切れる．なお，切れる概念はせん断応力ではない．粒子の位置の不連続性．骨がない筋肉だけの弾性体がぶよぶよ動いている感じ....



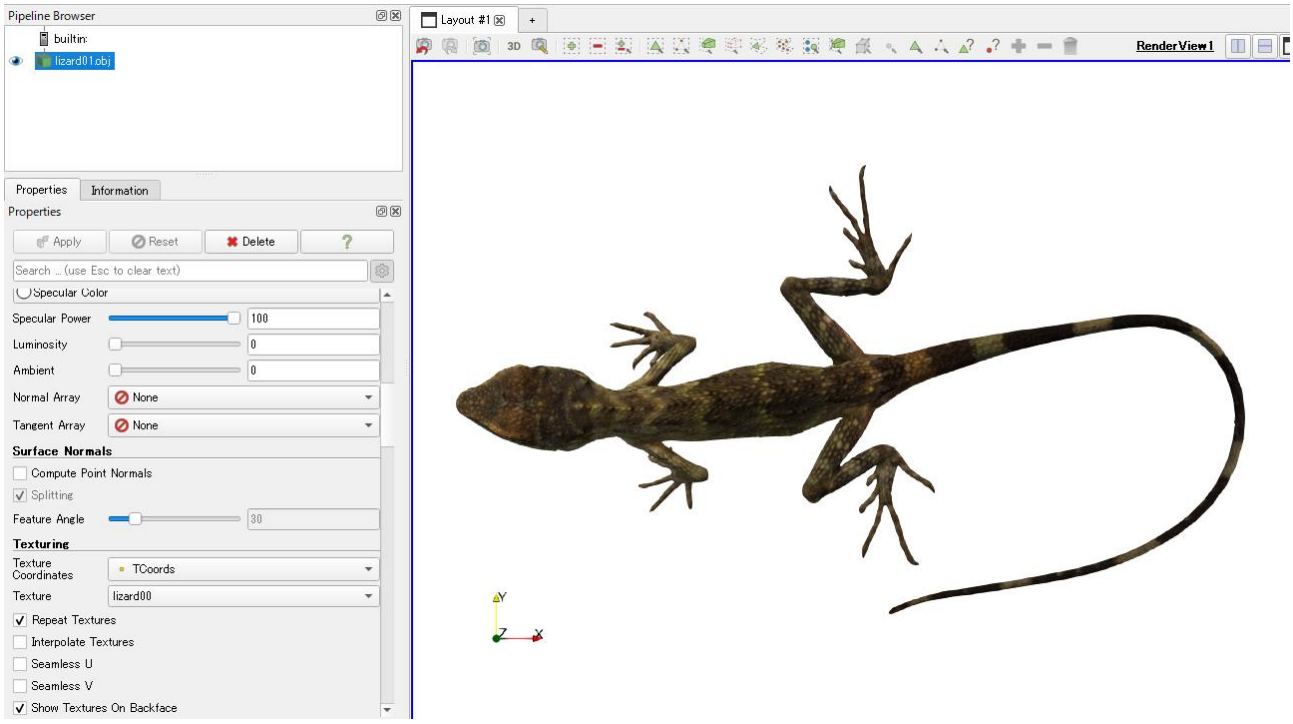
(3) 実行ファイル
FSIrun01.py：メインコード
motion.py：脚の運動
lizard01.obj：CAD の obj ファイル．uv データ付き
engine/MPM_solver.py: 力学計算ソルバ
lizard00.jpg：テクスチャ画像．こんな感じ．uv データはこの画像の画素になっており，点群の位置と対応している．

Taichi を activate して，ディレクトリを移動して python で実行する

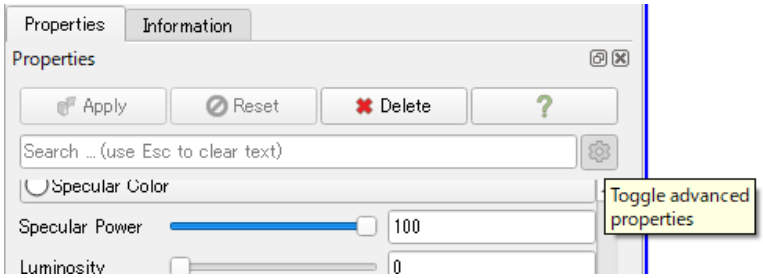
> source taichi_env/bin/activate
> cd /mnt/f/*********
> python3 FSIrun01.py

(4) **可視化**
(4-1) Taichi GUI の画像を ubuntu 上で動画化
ディスプレイの出力画面を保存した image/***.png ファイル群は，自動で動画にされる（out.mp4）．
なお，ここでも GPU の弊害があり，ファイルサイズが状況によって書き換えられている．コード上では，解像度 res=(1024, 760)に設定してあるが，100 回に 1 回ぐらい（1024, 758）になる．解像度が異なると動画にできないので補正している．いちいちコードが長くなる．

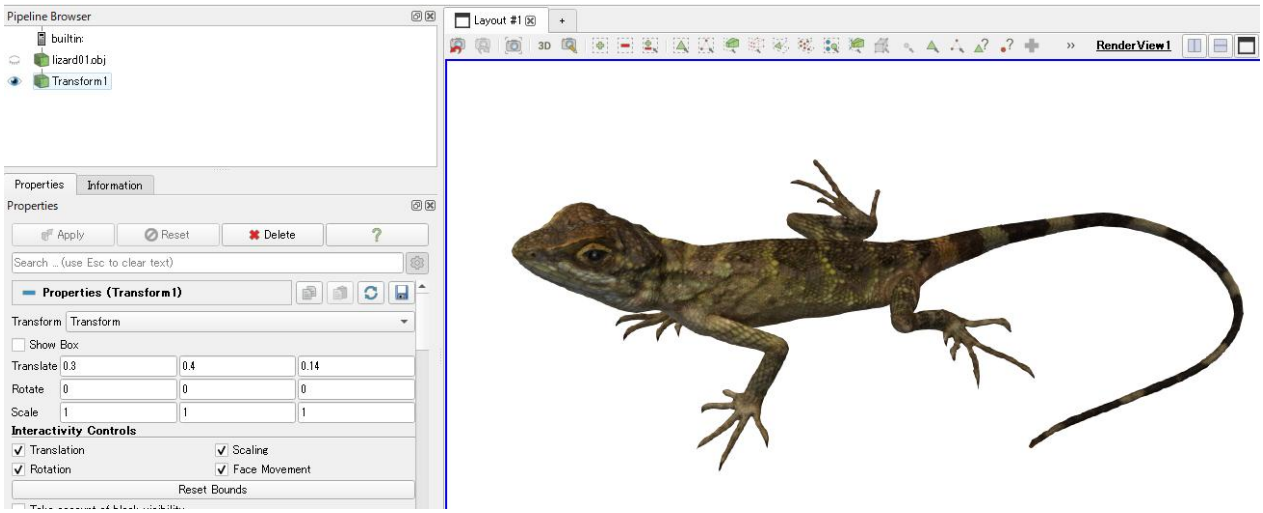
(4-2) Paraview で CG
　これとは別に，VTU ファイルを保存している．テクスチャを貼ってみる．
Paraview での流れ：
．テクスチャに対応した CAD の obj ファイル（画像ファイルとその uv 座標がある）を読み込む．
paraview の File->open から uv データ付き obj ファイル（lizard01.obj）を読み込む．
Display の Texuring の Texture Coordinates を「TCoords」にし，Texture の load で画像ファイル（lizard00.jpg）を読み込む．



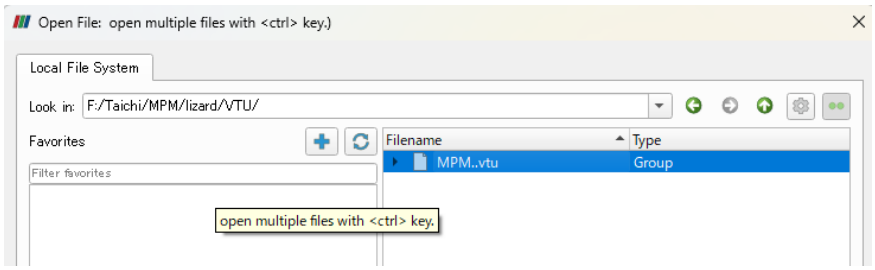
Texture が無い場合には，  をクリックすると出てくる．トグルになっている．



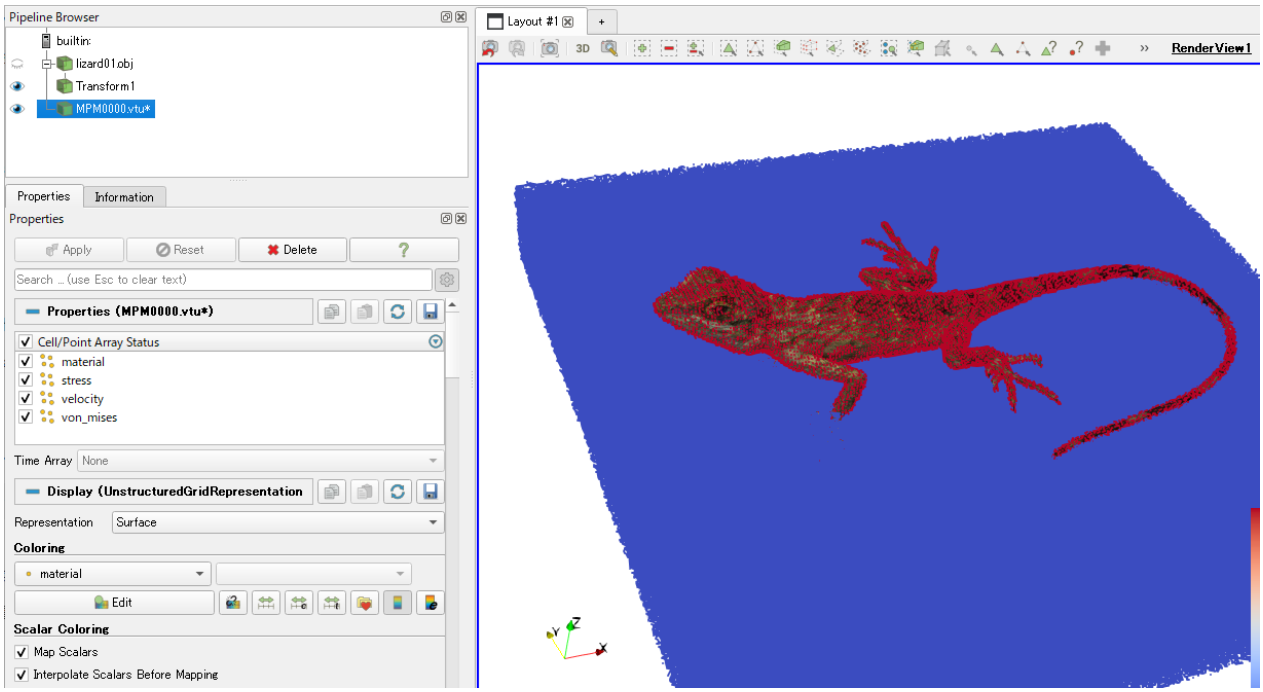
．シミュレーションの座標と CAD の座標が異なる場合には，Filters->Alphabetical->Transform で移動する．図は，(0.3, 0.4, 0.14)平行移動した例．上記同様 Texture も設定しておく．



．次に新たに File->Open から時系列の VTU ファイルを一度に読み込み，Apply する．

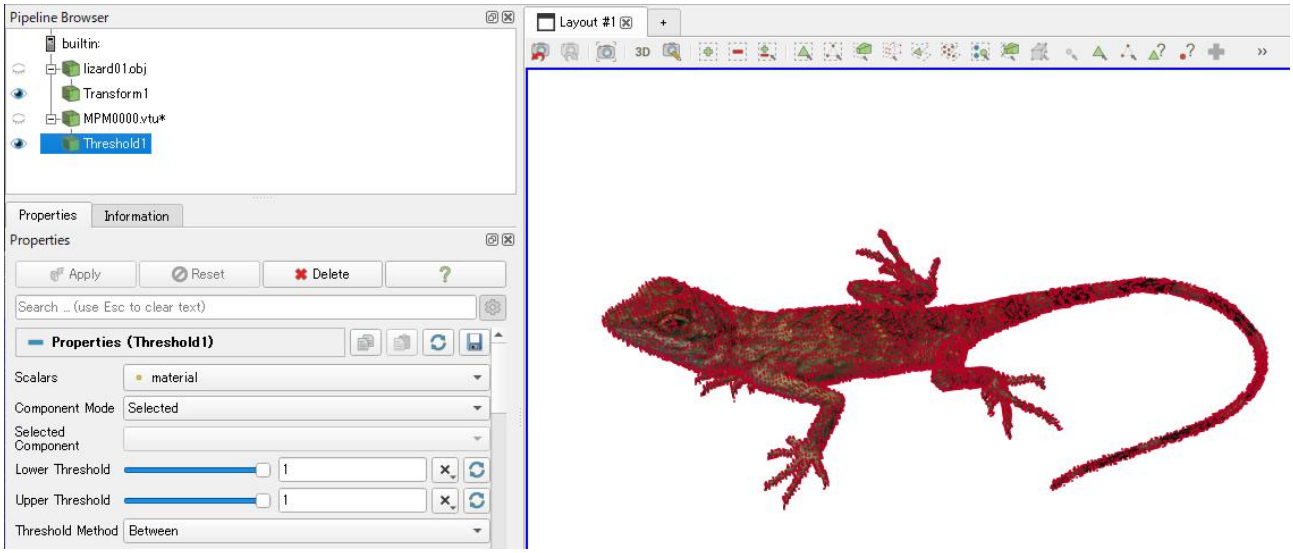


カラーリングを material にすると，弾性体粒子（赤）と obj ファイルが一致していることが確認できる．

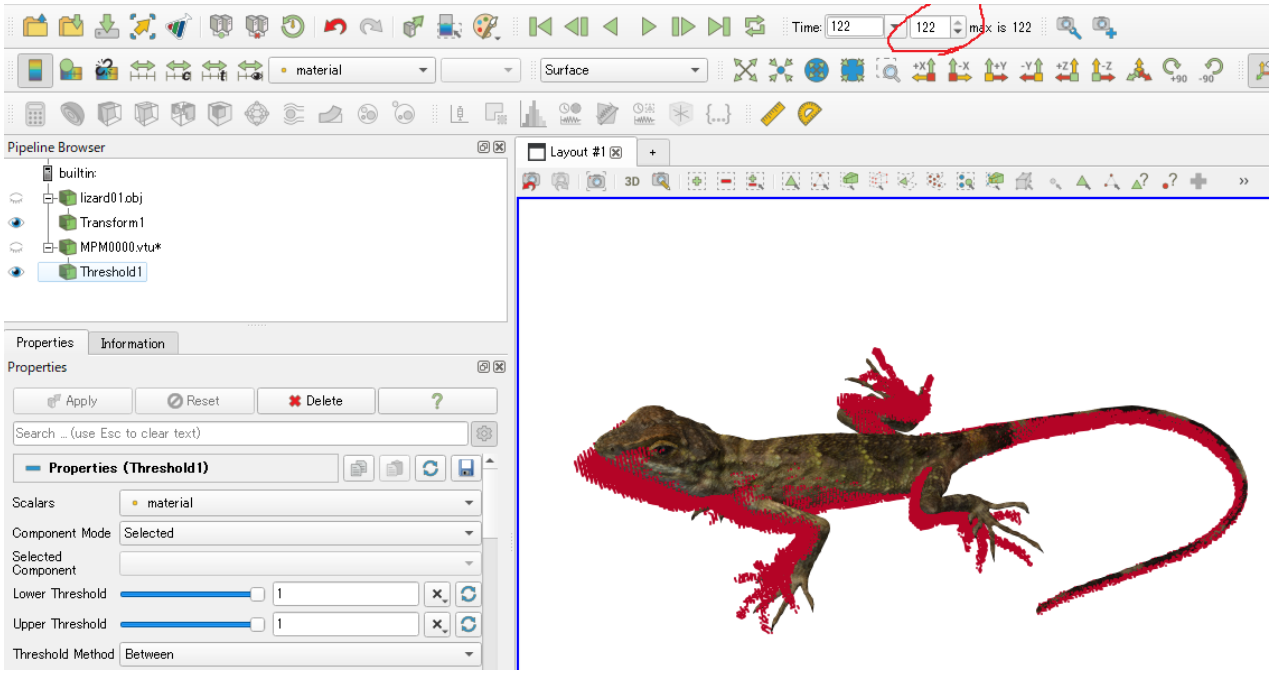


．次に，初期弾性体粒子位置（t=0）と現在の時刻 t の間の粒子の移動量を U として生成し，obj の点データをこれに基づき移動させ，そこにテクスチャを張り付ける．

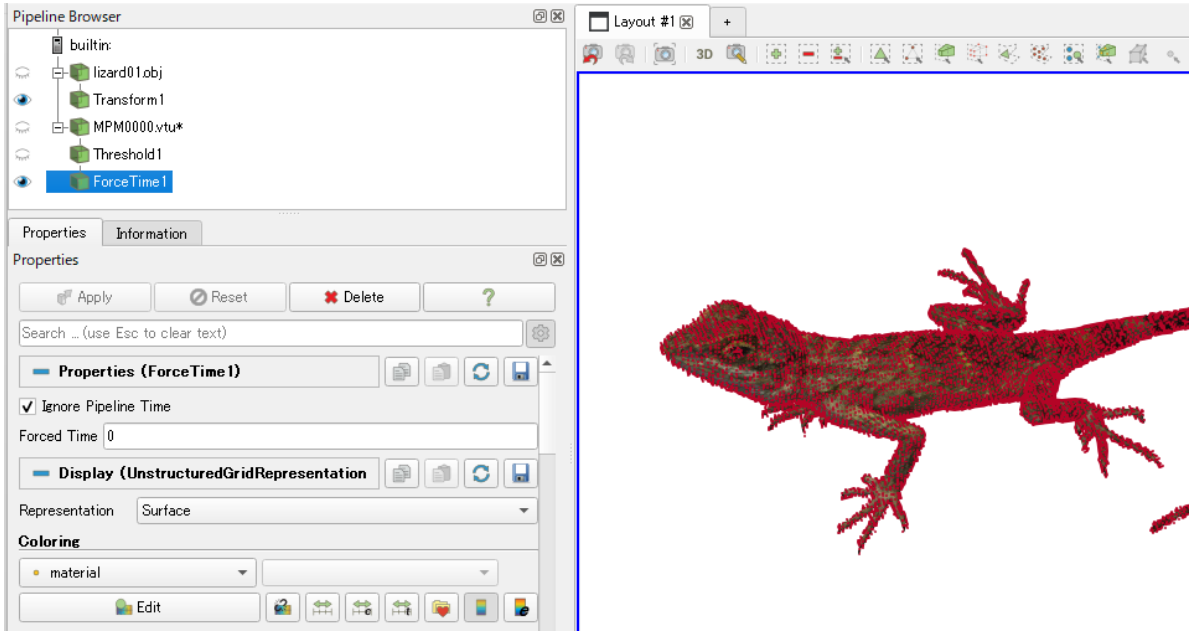
まず，Filters->Alphabetical->Threshold で，Lower Threshold=Upper Threshold=1 にして弾性体粒子（material=1）を抽出する．



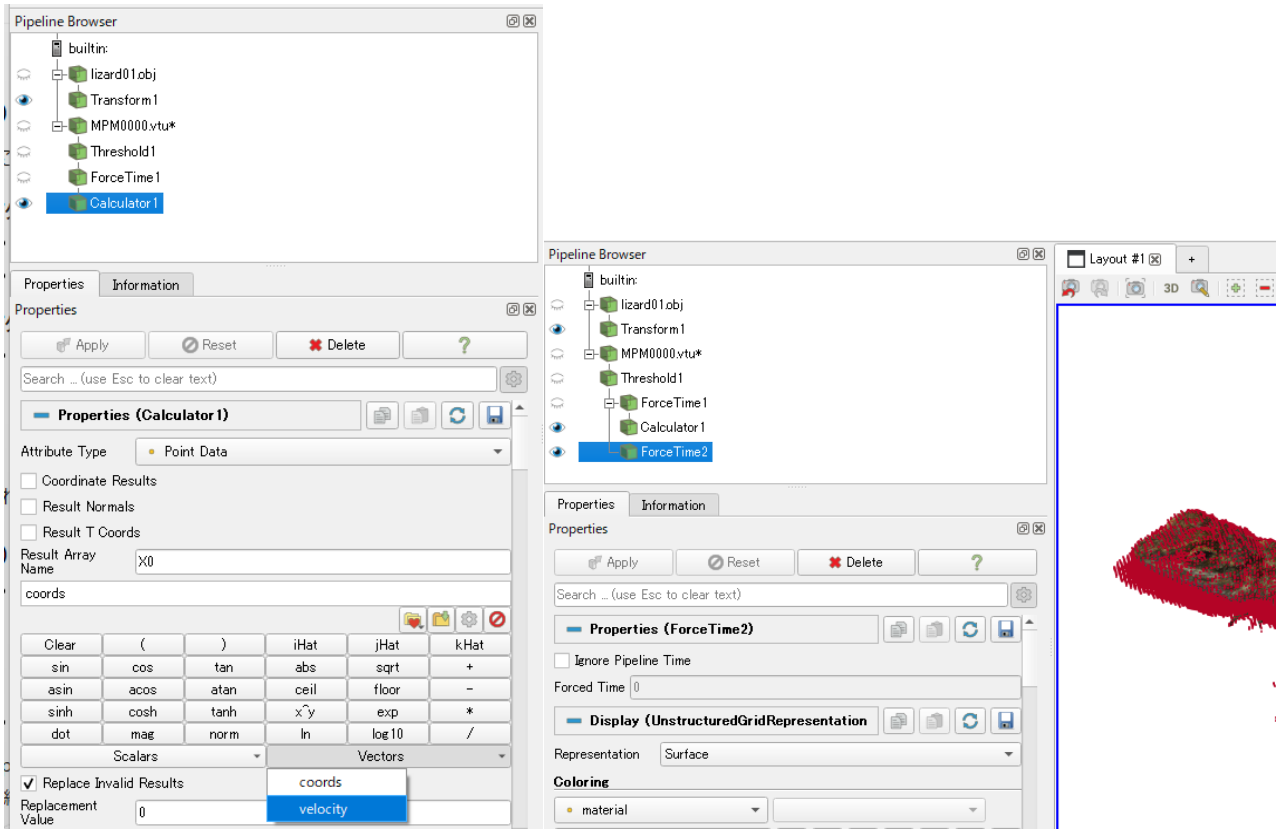
次に時刻を max に，粒子が移動していることを確認する．



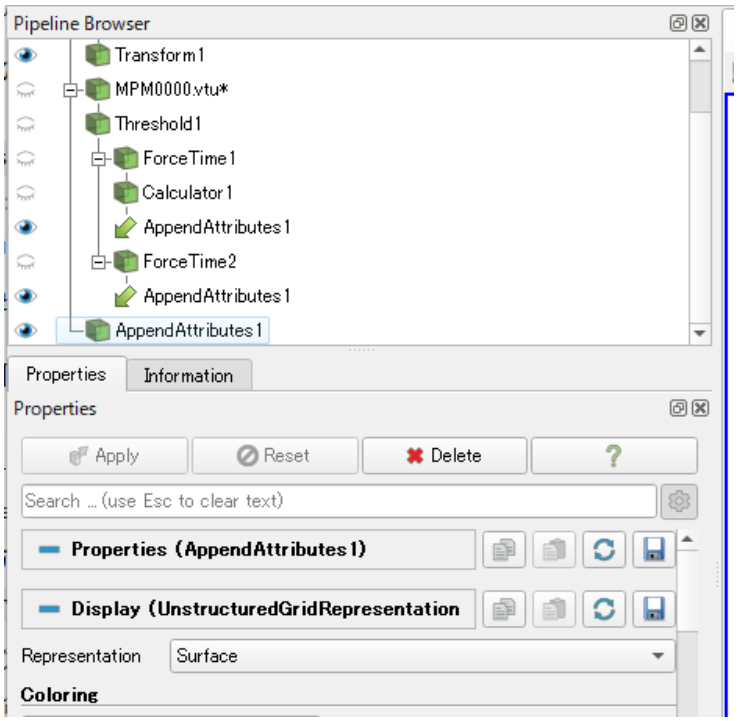
次に，Filters->Alphabetical->Force Time で，Forced Time=0, Ignore Pipleline Time にチェックして Apply する．t=0 番目の粒子データが表示される．時間を ignore したので，Time の設定に関係なく t=0 のデータが返される．



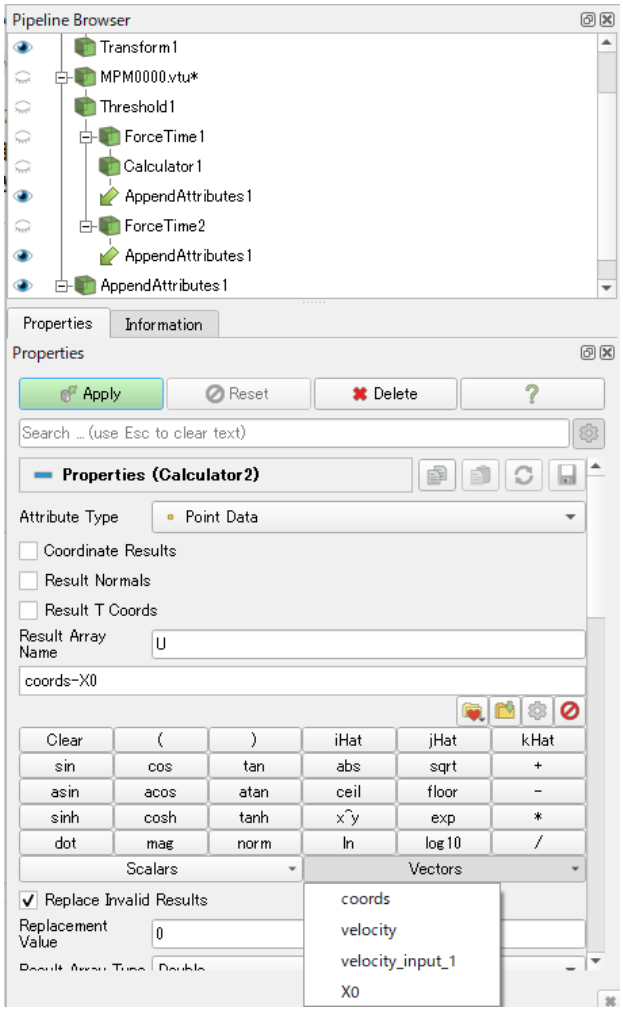
Fileters->Alphabetical->Calculator で Result Array Name を”X0”に設定し，Vectors から”coords”を選択し，Apply する．X0 が弾性体粒子の初期位置になる．次に，pipeline の Threshold1 を選択した状態で，Filters->Alphabetical->Force Time を選択し，Ignore pipeline Time のチェックを外し（t=Time のデータを読み込むようにし），Apply する．



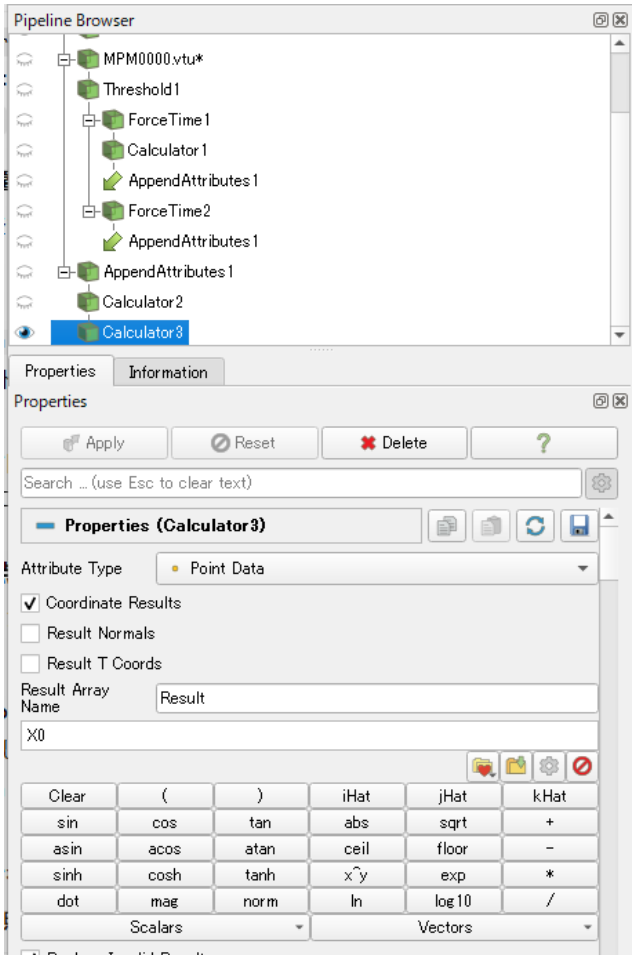
次に，Calculator1（X0 データ）と Force Time2 を CTL を押しながら二つ選択し，二つが選択されている状態で，Filters->Alphabetical->Append Attributes でデータを結合する．



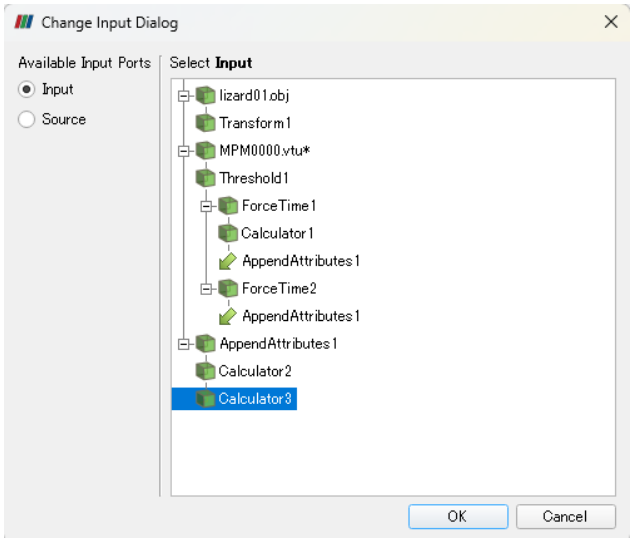
次に，Append Attributes を選択して Filters->Alphabetical->Calculator で，Result Array Name を”U”にし，Vector で”coords”を選択，”-”をクリック，Vector で”X0”を選択して
U＝coords－X0
の変位ベクトルデータを生成する．

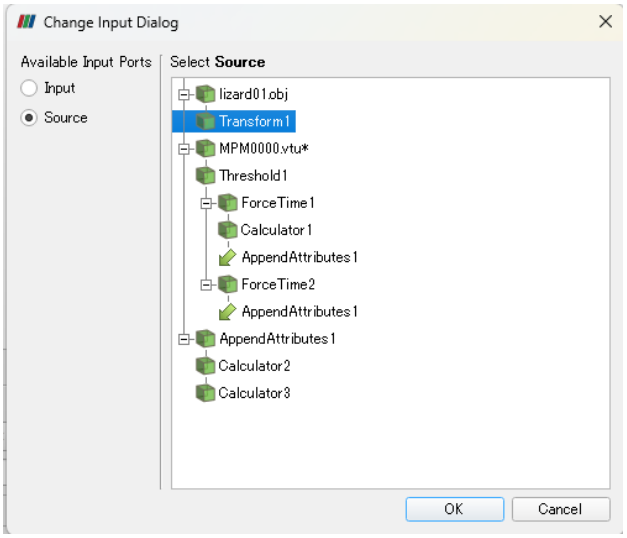


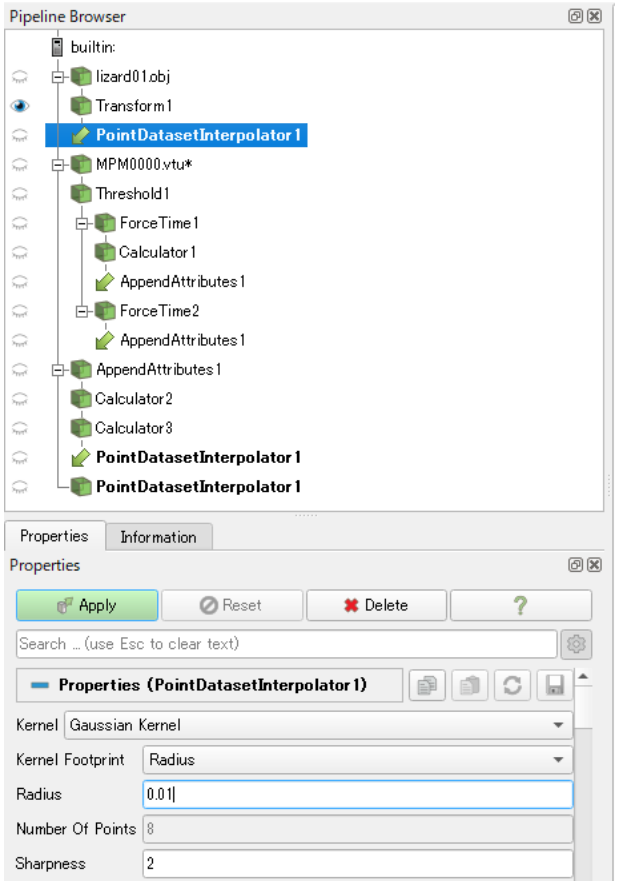
最後にこれを座標データにする．Filters->Alphabetical->Calculator で，Coordinate Results をチェックし，Vectors から”X0”を選択し，Result Array Name を適当に”X0cordinate”にし，Apply する．陽に使わないのでなんでもいい．



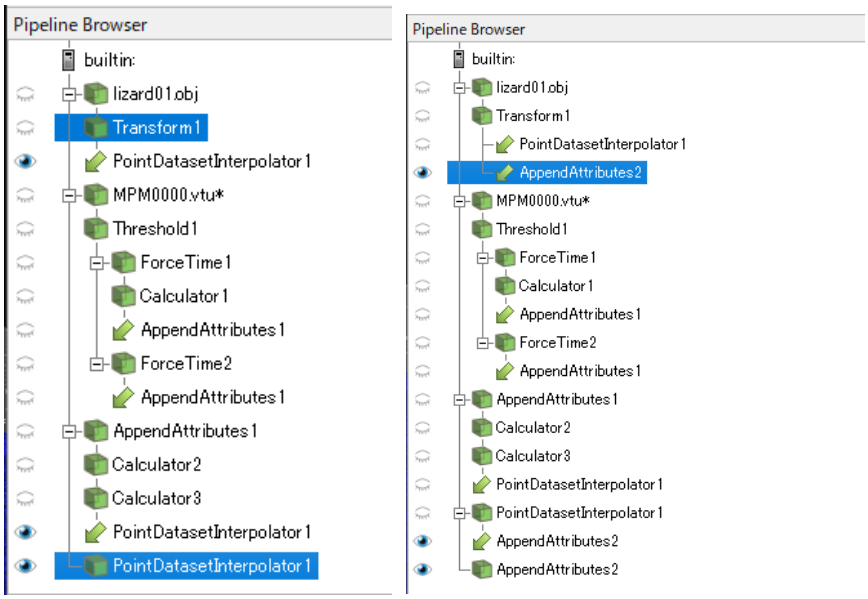
．次に，座標にした Xcoordinates を使って，obj の点データを対応図づける．Calculator3 を選択した状態で，Filters->Alphabetical->Point Dataset Interpolator を選択し，Input を Calculator3（X0coordinates），Source を Transform1（変換後の obj の点群データ）を選択して，OK する．その後，Gaussian Kernel を選択し，半径を 0.01m（粒子径の 2 倍から 3 倍程度）にし，Apply する．



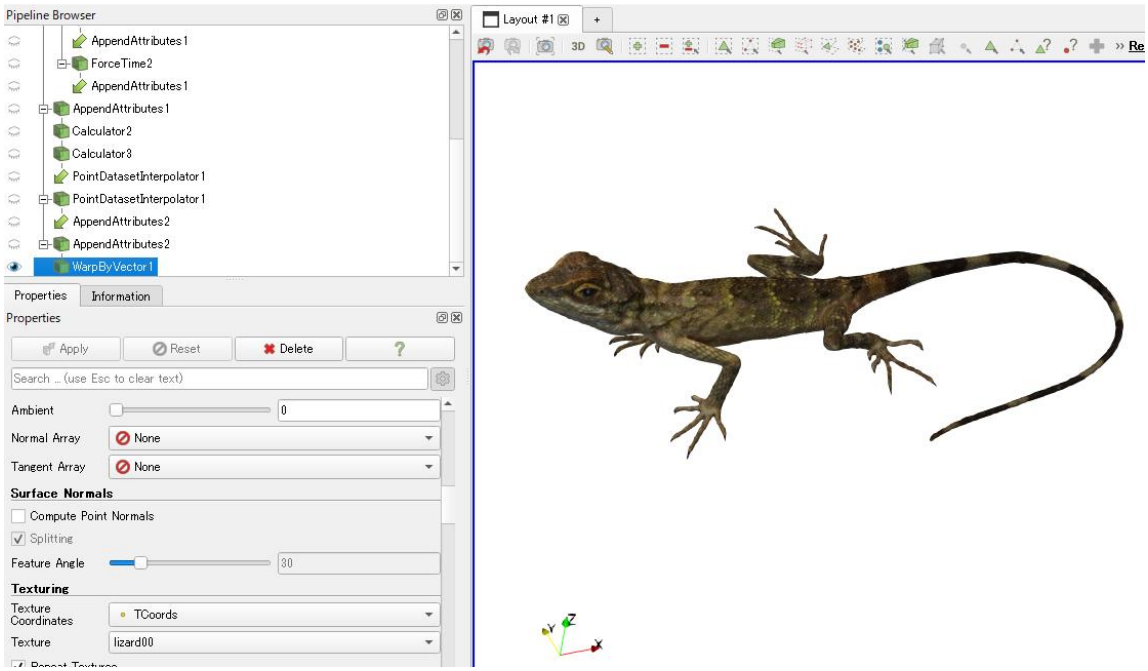




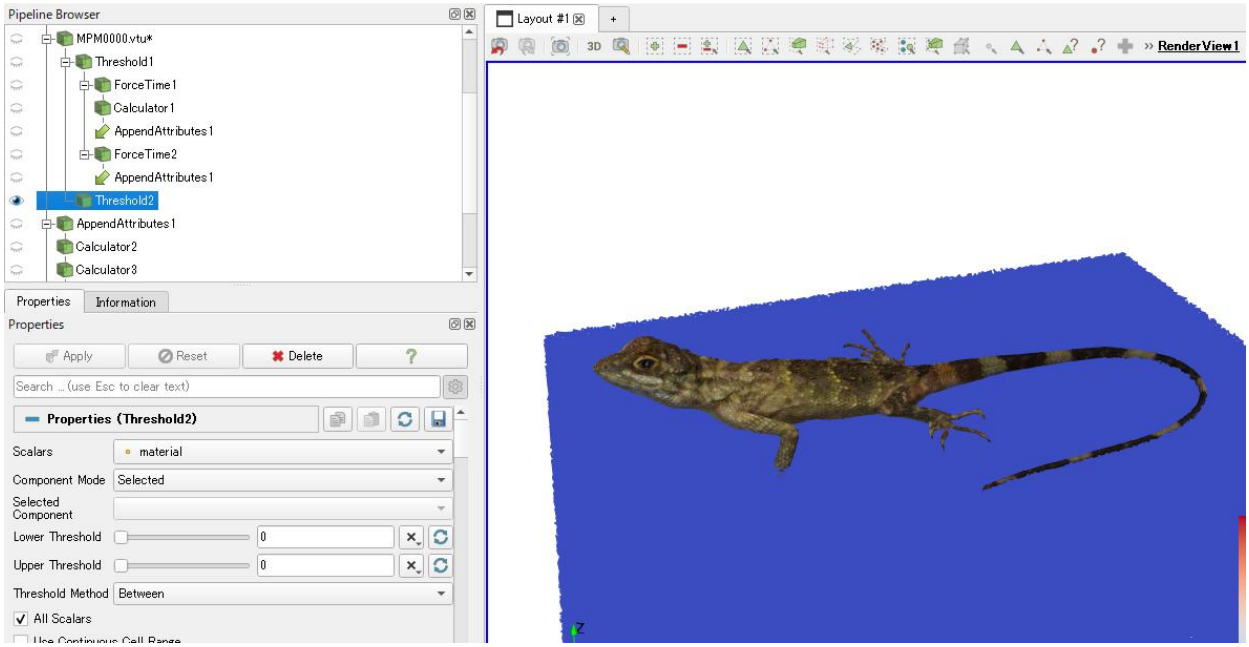
次に，テクスチャを貼るための処理を行う．やらなくていい作用であるはずだが，Texture の TCoords を選択できない現象が起こるので念のため．
TCoords が存在している Transform1（平行移動後の obj 位置情報）と，PointDataSetInterpolator1（obj と初期粒子位置を結合したデータ）を ctl を押しながら同時に選択し，Filters->Alphabetical->Append Attribute を選択し，Apply する．



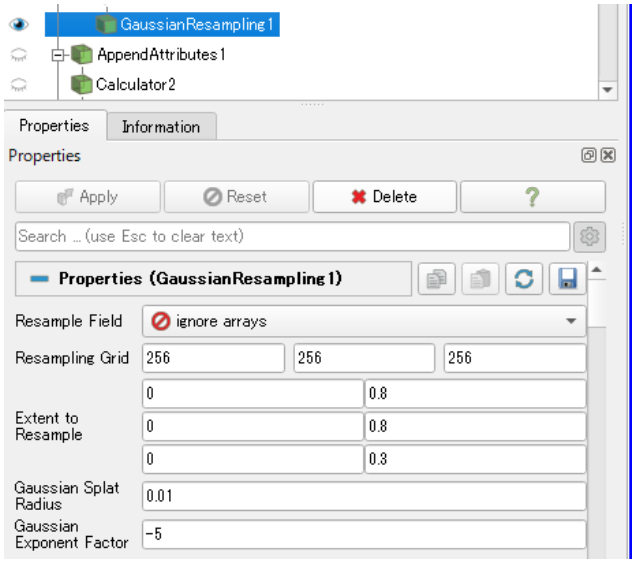
最後に AppendAttributes2 を選択し，Filters->Alphabetical->Warp by Vector を選択し，Vectors に粒子の移動量 U を選択，Texture で元の画像を選択して Apply する．



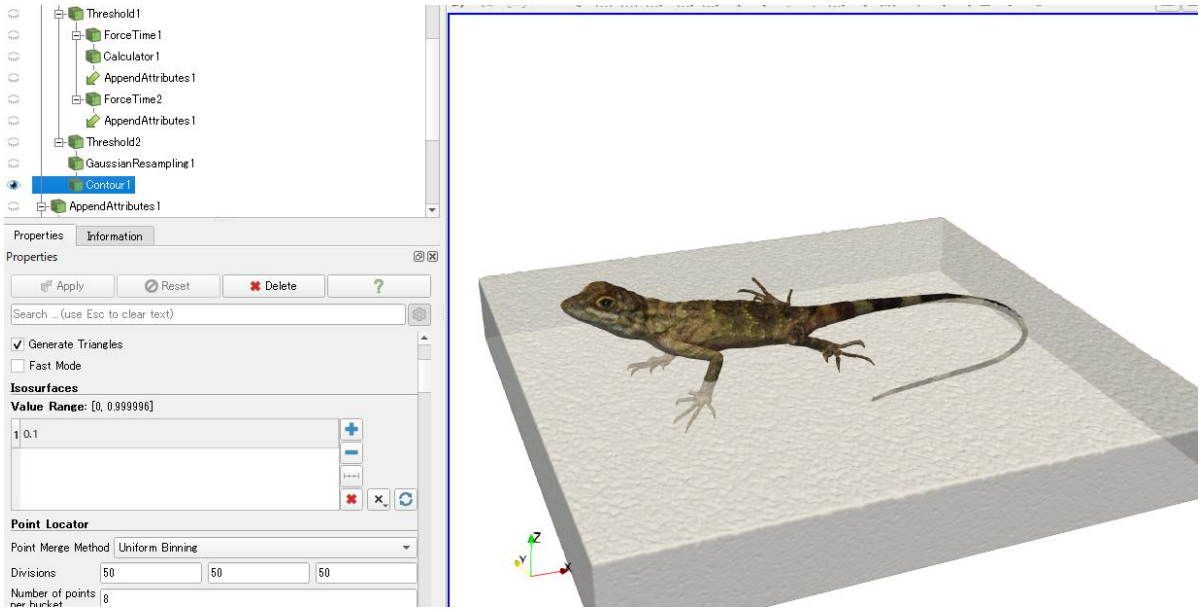
．次に水面を描画する．pipeline から時系列データを選択し，Filters->Alphabetical->Threshold で Lower and Upper Threshold=0 にして（material=0 が水）Apply すると，水面が描画される．



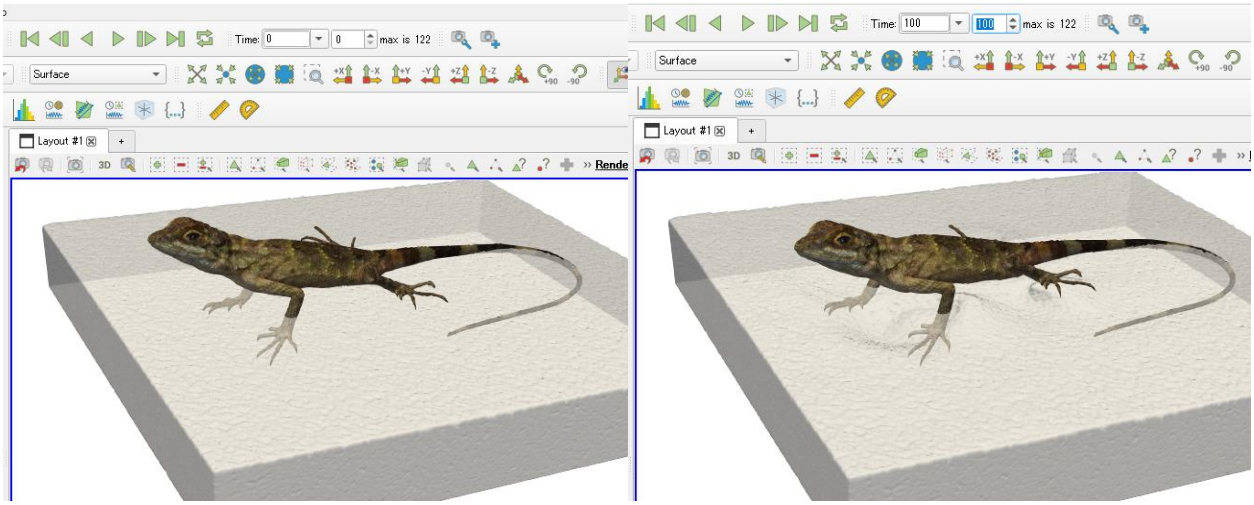
．同様に，Gaussian Resampling する．



Contour で value range を 0.1 程度にし，Opacity を 0.6 程度にすると図のようになる．



時刻を進めるとテクスチャが時系列で変形していることがわかる．


Time: 0
0
max is 122
Surface
Layout #1
Time: 100
100
max is 122
Surface
Layout #1