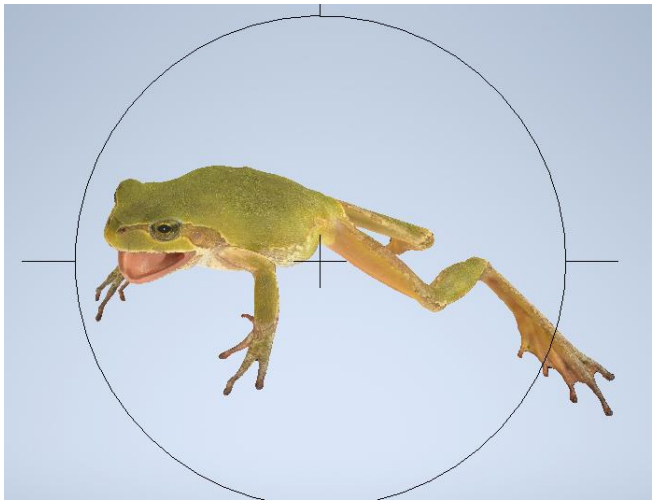
Frog Diving by MLS-MPM in Taichi

11 Mar. 2026

もうただの CG．弾性体のカエルに初速度与えて水に落としただけ．詳細は lizard を参照.

計算空間：0.2m*0.2m*0.2m
格子解像度：128*128*128→格子サイズ 0.2/128=1.55 [mm]
粒子サイズ：1 格子に 2^3=8 つ
水領域：0.2m*0.2m*0.05m
カエル：弾性体粒子，体長 0.04m ぐらい
走行：初速度（-2, 0, -4） [m/s]．高さ 1m あたりから水に飛び込んだカエル.
標本データはここ：
https://sketchfab.com/3d-models/cc0-japanese-tree-frog-3597ee5176404efc8e8d78affe0fd223



座標系が異なるので，z 軸を重力方向に回転させたり，サイズ変更，平行移動のコードは以下．適当に修正する.

```
import sys

input_file = "frog_removed_square.obj"                  # 読み込みファイル．CAD の obj
output_file = "frog.obj"                                # 変換後の出力ファイル

with open(input_file, "r") as f, open(output_file, "w") as out:
    for line in f:
        if line.startswith("v "):
            parts = line.split()
            x = float(parts[1])                         # x, y, z が元の点群座標
            y = float(parts[2])
            z = float(parts[3])

            # x 軸 +90° 回転（Y→Z）                    x2, y2, z2 が変換後の点群座標
            x2 = x
            y2 =-z
```

```
                z2 = y

                # 平行移動
                #x2 *= 0.1
                #y2 *= 0.1
                # z2 += 0.5

                # スケール 1/10
                #x2 *= 0.1
                #y2 *= 0.1
                #z2 *= 0.1

                out.write(f"v {x2} {y2} {z2}¥n")
            else:
                out.write(line)

print("done:", output_file)
```

(1) 実行ファイル

frog.py：メインコード

frog.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

frog.jpg：テクスチャ画像.

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 frog.py
```

(2) **可視化**

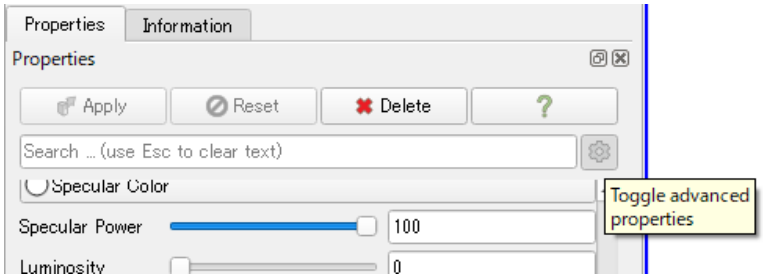
Paraview での流れ：

．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.

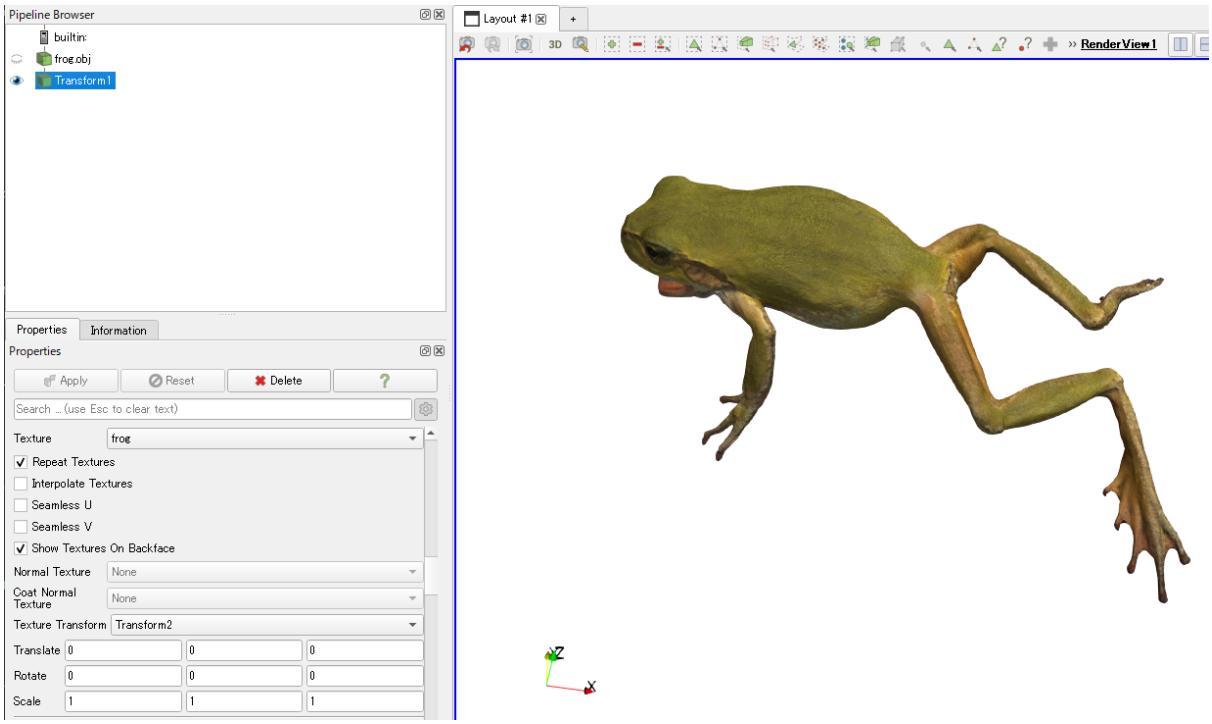
paraview の File->open から uv データ付き obj ファイル（frog.obj）を読み込む.

Display の Texuring の Texture Coordinates を「TCoords」にし，Texture の load で画像ファイル（frog.jpg）を読み込む.

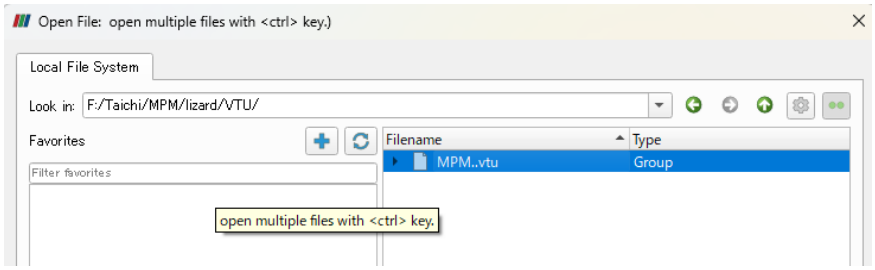
Texture が無い場合には，  をクリックすると出てくる．トグルになっている.



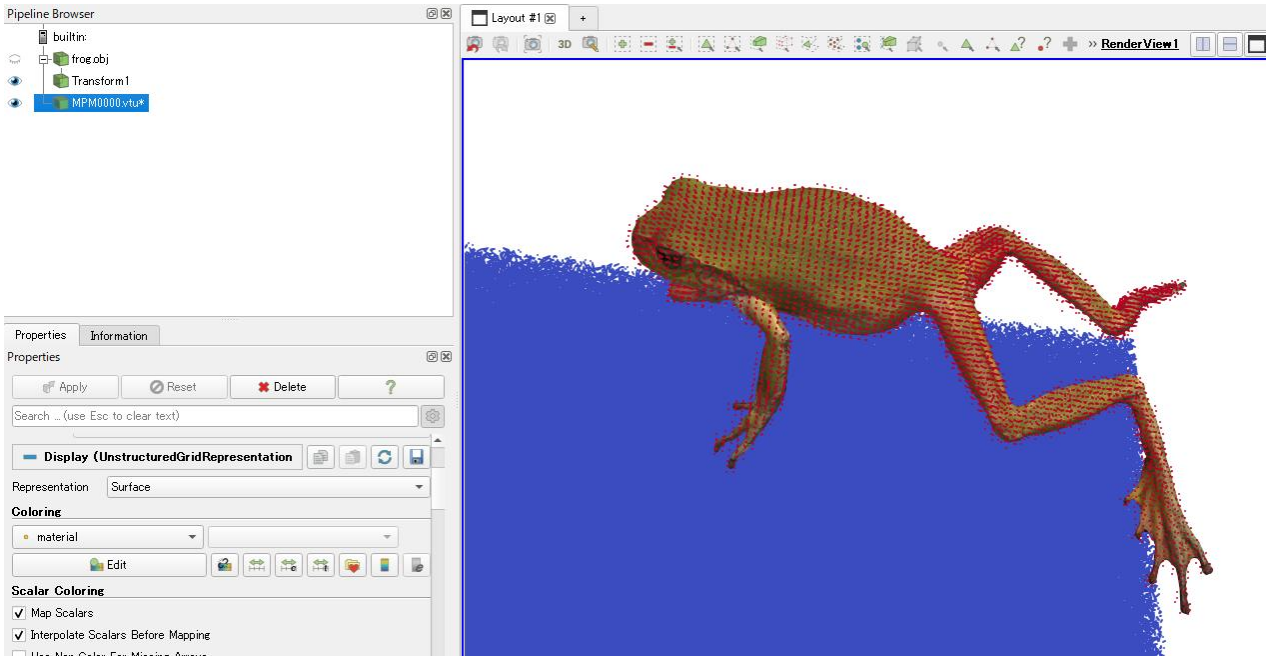
．シミュレーションの座標と CAD の座標が異なる場合には，Filters->Alphabetical->Transform で移動する．図は，(0.13, 0.1, 0.15)平行移動した例．上記同様 Texture も設定しておく．



．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．

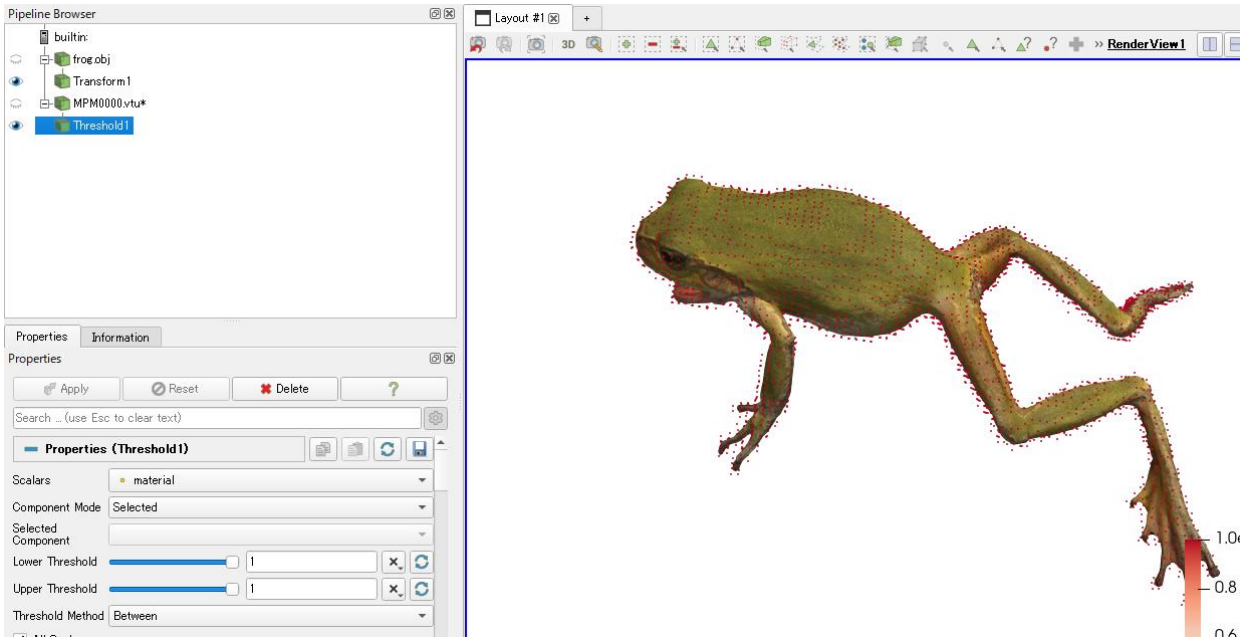


カラーリングを material にすると，弾性体粒子（赤）と obj ファイルが一致していることが確認できる．

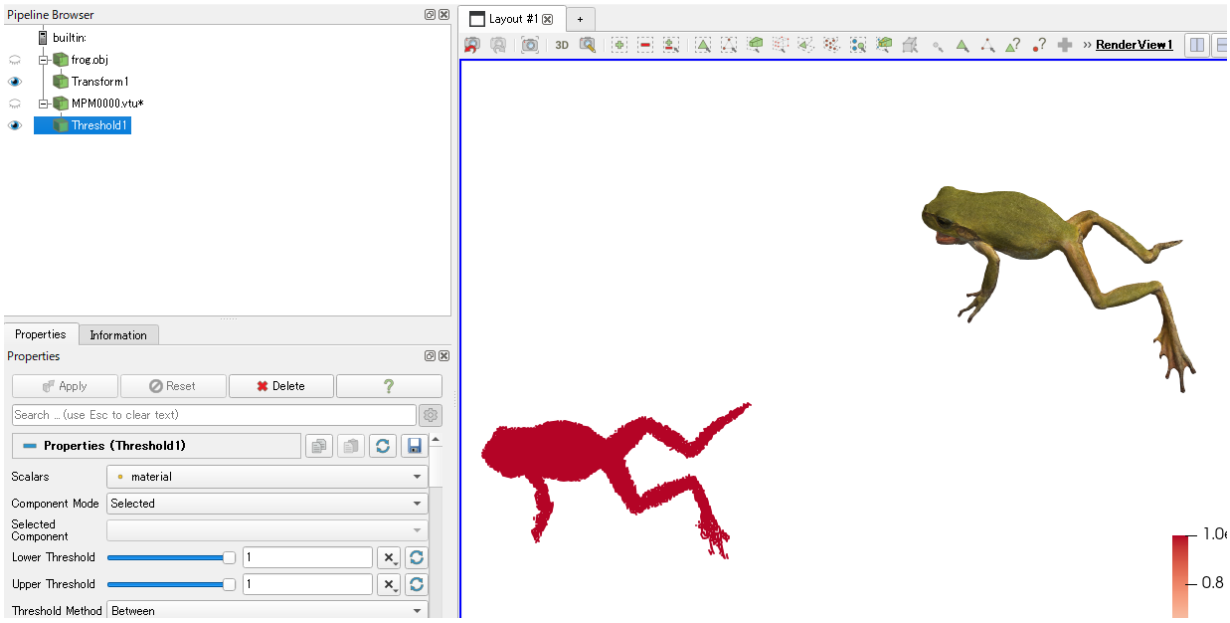


．次に，初期弾性体粒子位置（t=0）と現在の時刻 t の間の粒子の移動量を U として生成し，obj の点データをこれに基づき移動させ，そこにテクスチャを張り付ける．

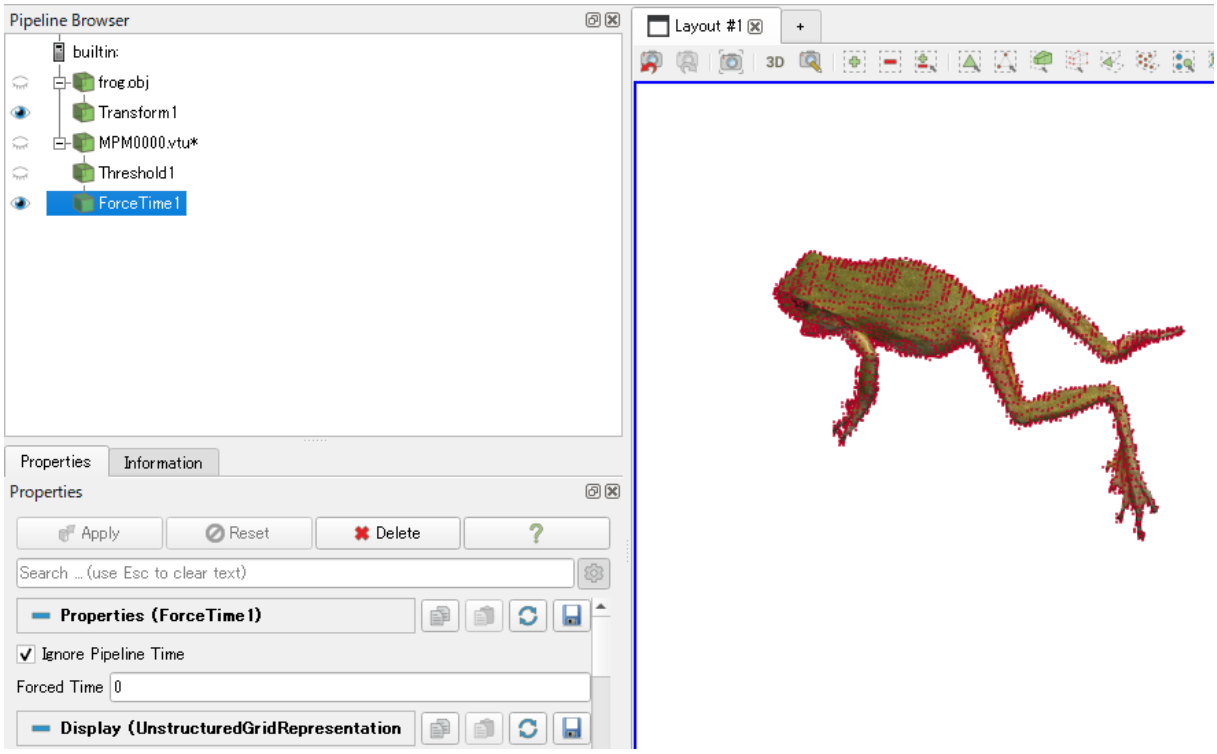
まず，Filters->Alphabetical->Threshold で，Lower Threshold=Upper Threshold=1 にして弾性体粒子（material=1）を抽出する．



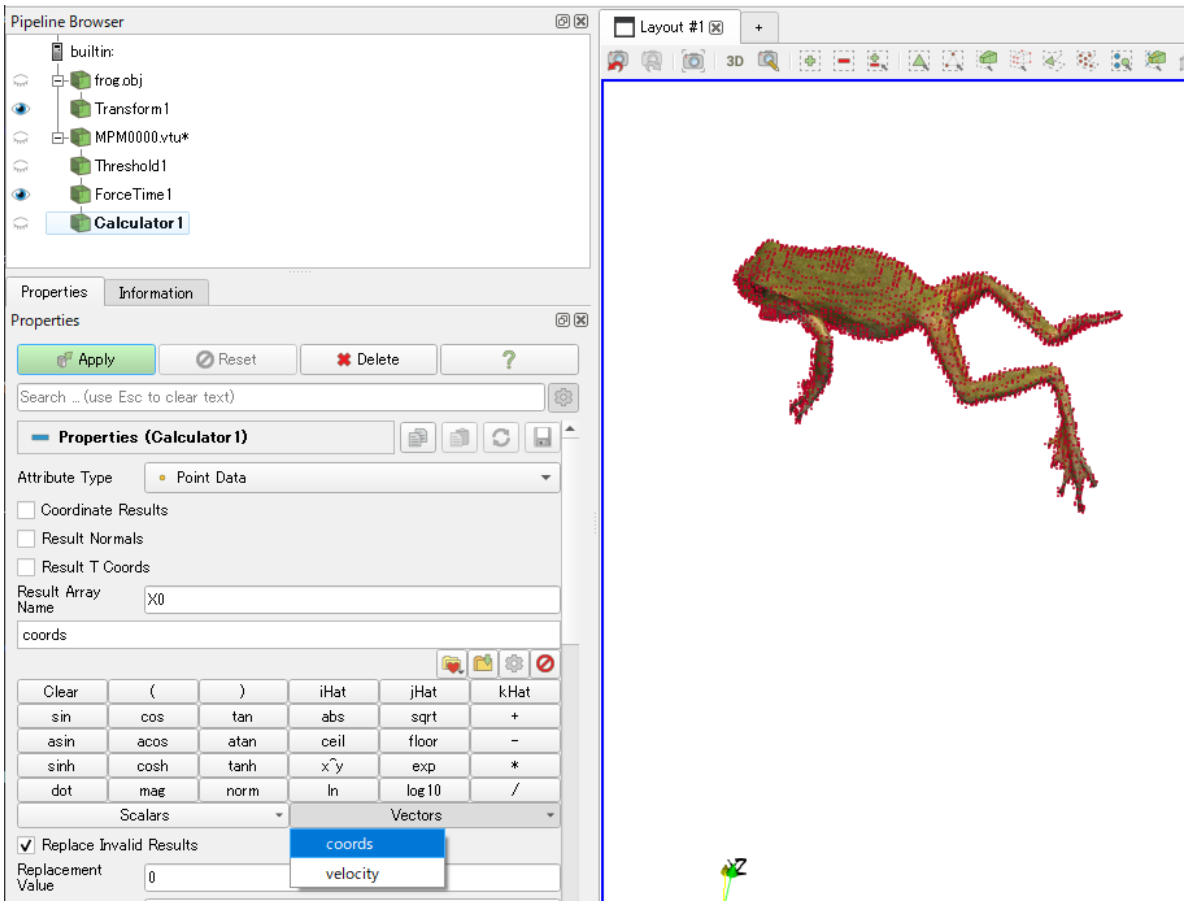
次に時刻を max に，粒子が移動していることを確認する．



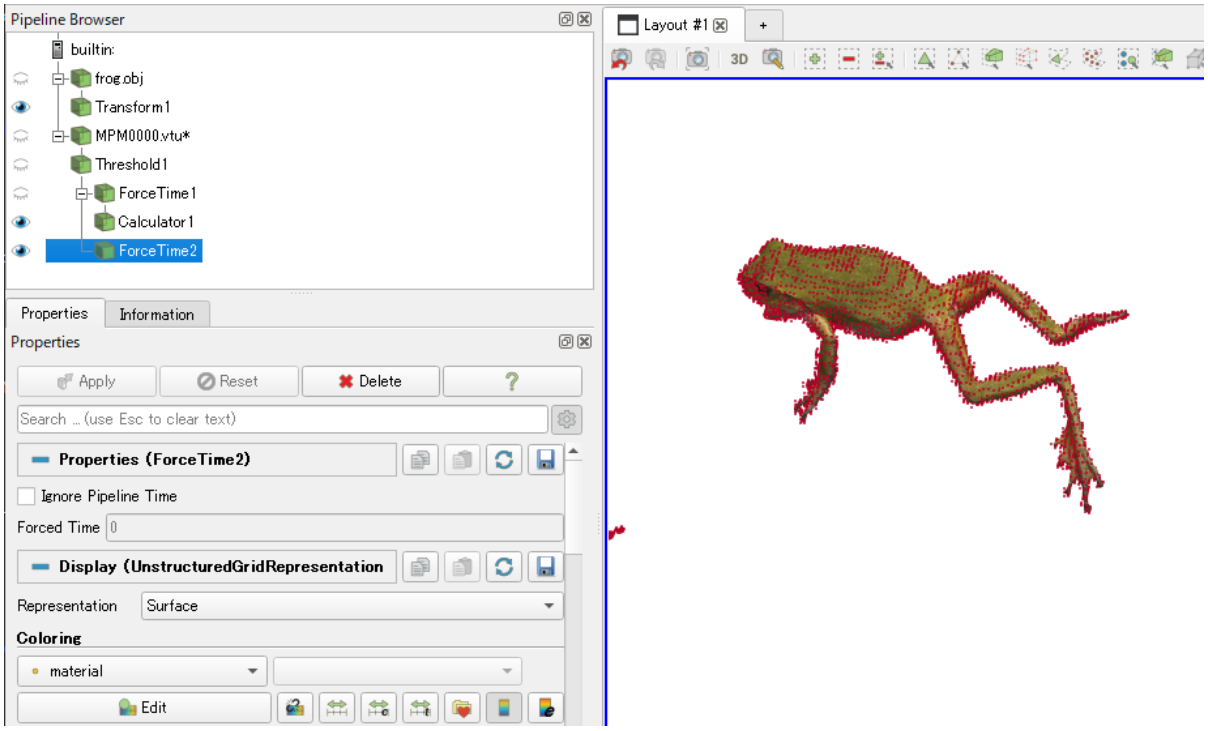
次に，Filters->Alphabetical->Force Time で，Forced Time=0, Ignore Pipleline Time にチェックして Apply する．t=0 番目の粒子データが表示される．時間を ignore したので，Time の設定に関係なく t=0 のデータが返される．



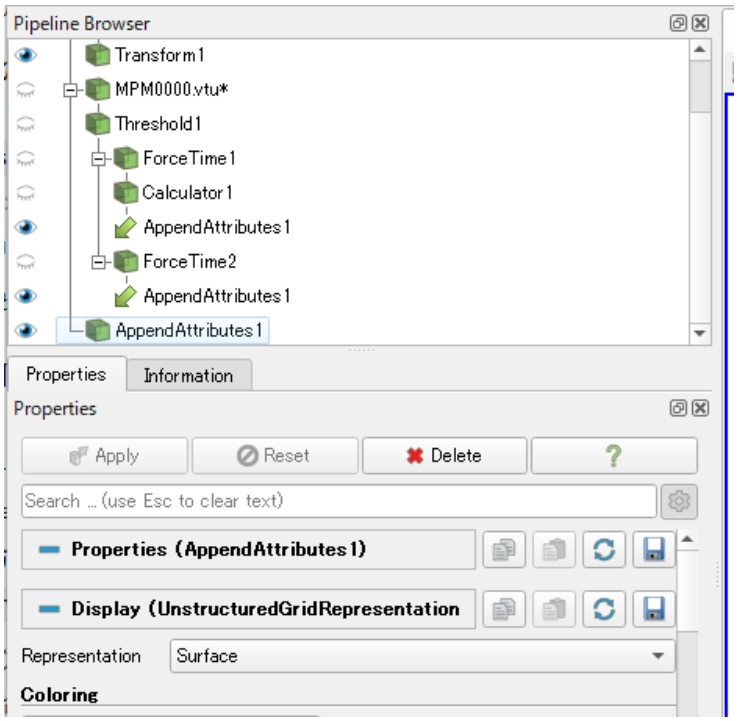
Fileters->Alphabetical->Calculator で Result Array Name を”X0”に設定し，Vectors から”coords”を選択し，Apply する．X0 が弾性体粒子の初期位置になる．



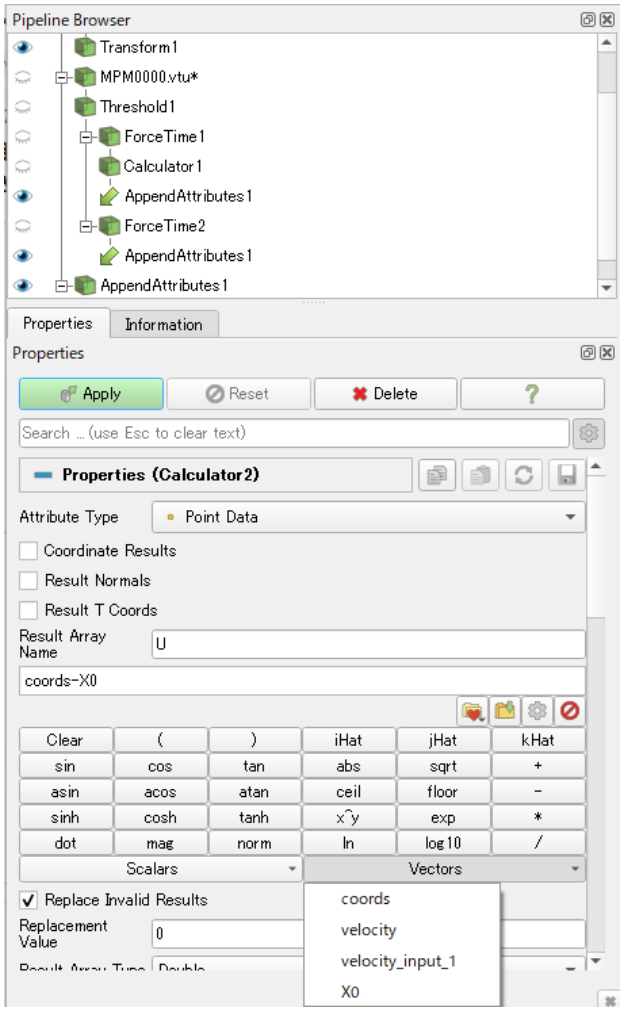
次に，pipeline の Threshold1 を選択した状態で，Filters->Alphabetical->Force Time を選択し，Ignore pipeline Time のチェックを外し（t=Time のデータを読み込むようにし），Apply する．



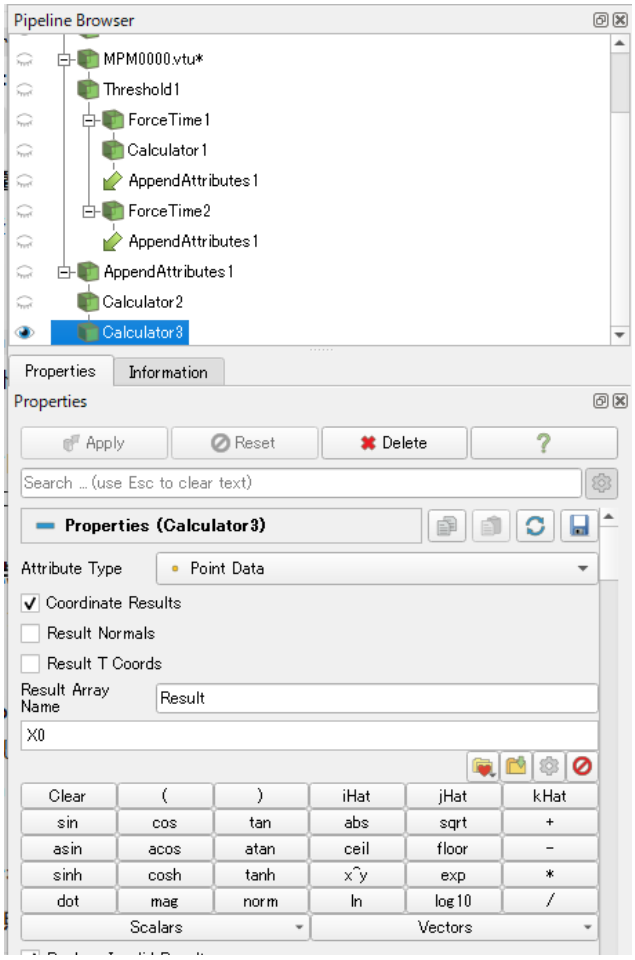
次に，Calculator1（X0 データ）と Force Time2 を CTL を押しながら二つ選択し，二つが選択されている状態で，Filters->Alphabetical->Append Attributes でデータを結合する．



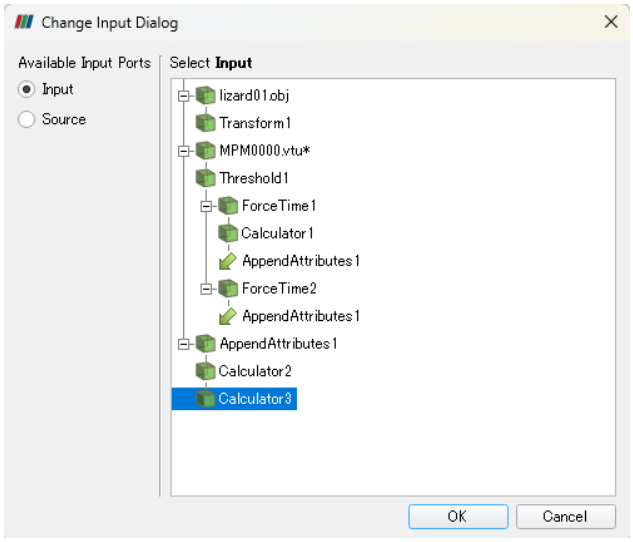
次に，Append Attributes を選択して Filters->Alphabetical->Calculator で，Result Array Name を”U”にし，Vector で”coords”を選択，”-”をクリック，Vector で”X0”を選択して
U＝coords－X0
の変位ベクトルデータを生成する．

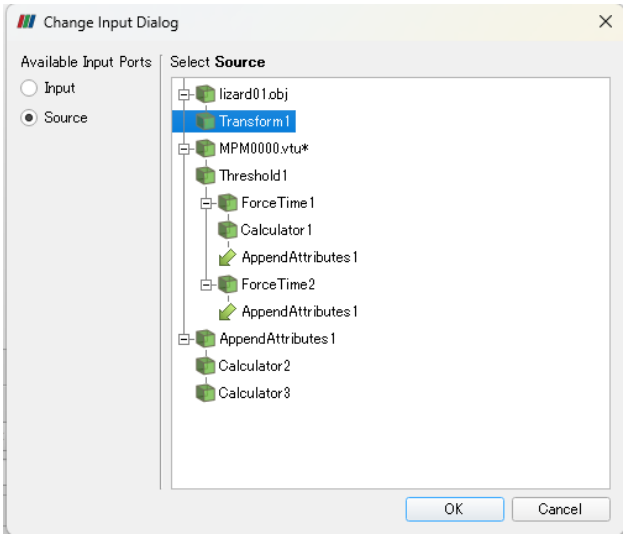


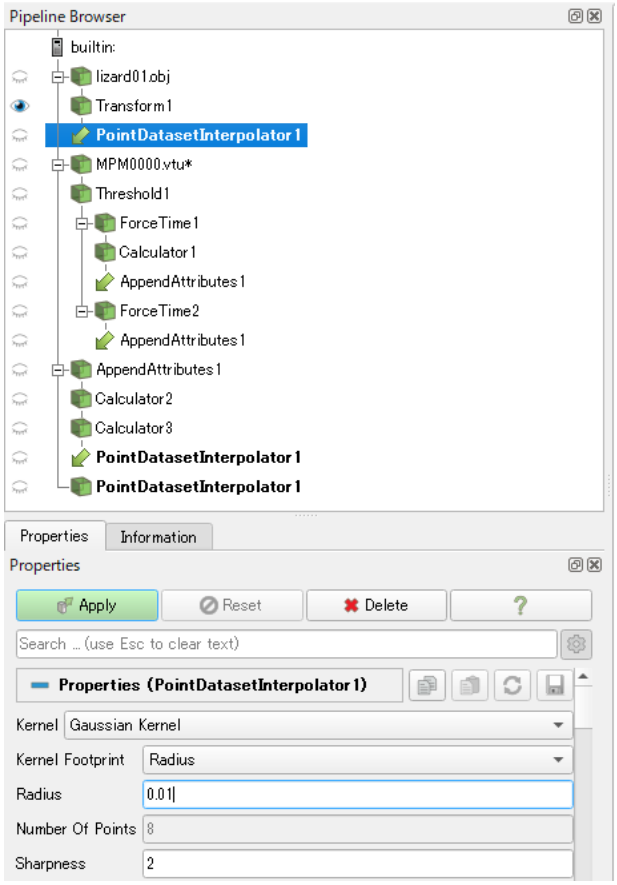
最後にこれを座標データにする．Filters->Alphabetical->Calculator で，Coordinate Results をチェックし，Vectors から”X0”を選択し，Result Array Name を適当に”X0cordinate”にし，Apply する．陽に使わないのでなんでもいい．



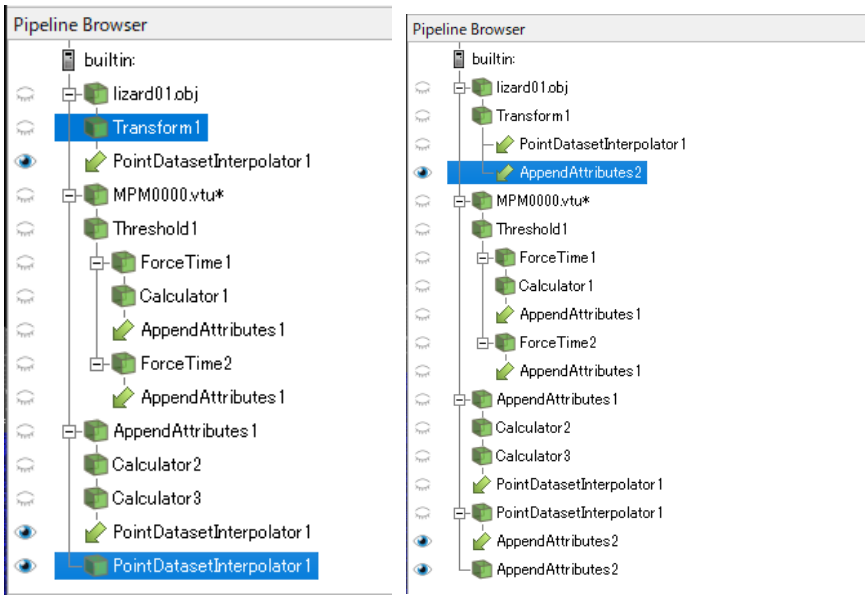
．次に，座標にした Xcoordinates を使って，obj の点データを対応図づける．Calculator3 を選択した状態で，Filters->Alphabetical->Point Dataset Interpolator を選択し，Input を Calculator3（X0coordinates），Source を Transform1（変換後の obj の点群データ）を選択して，OK する．その後，Gaussian Kernel を選択し，半径を 0.005m（粒子径の 2 倍から 3 倍程度）にし，Apply する．



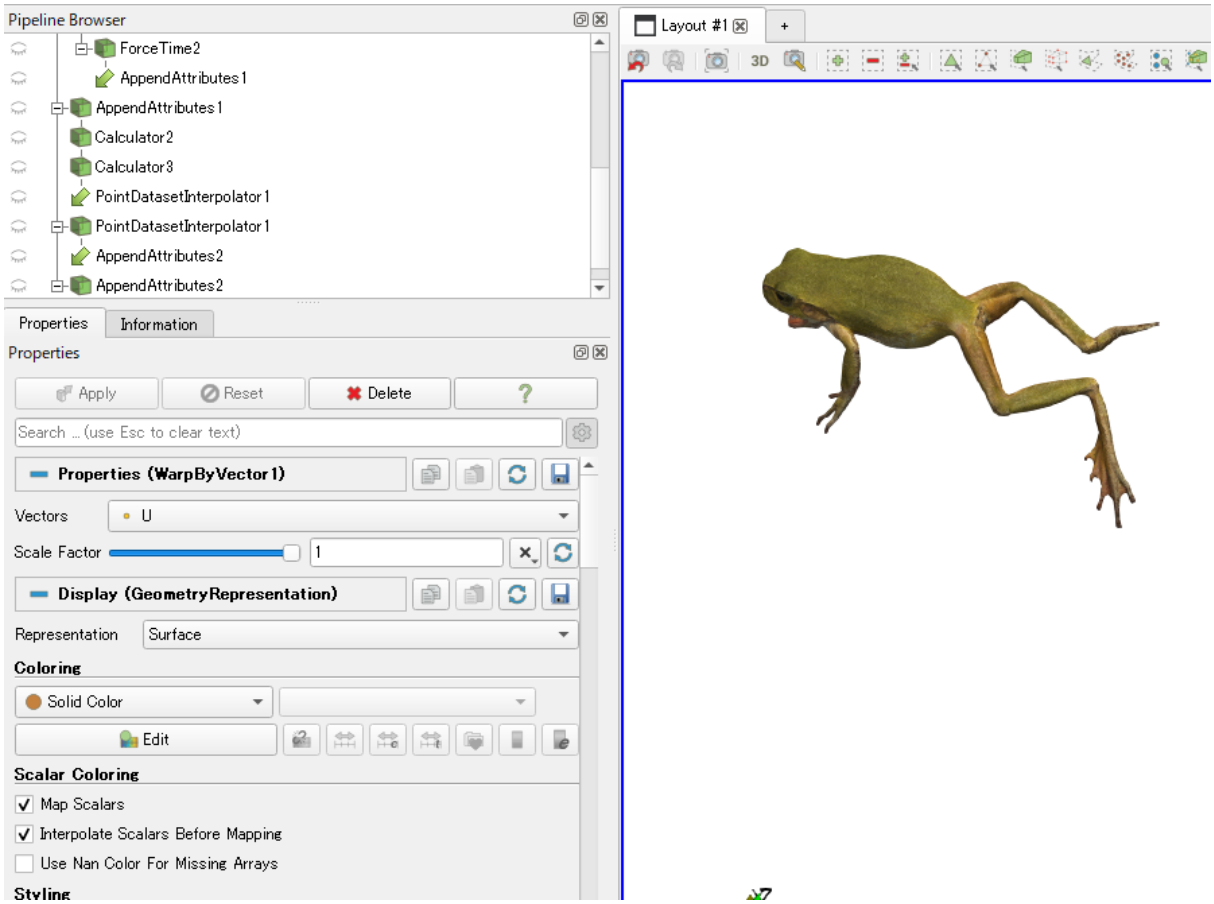




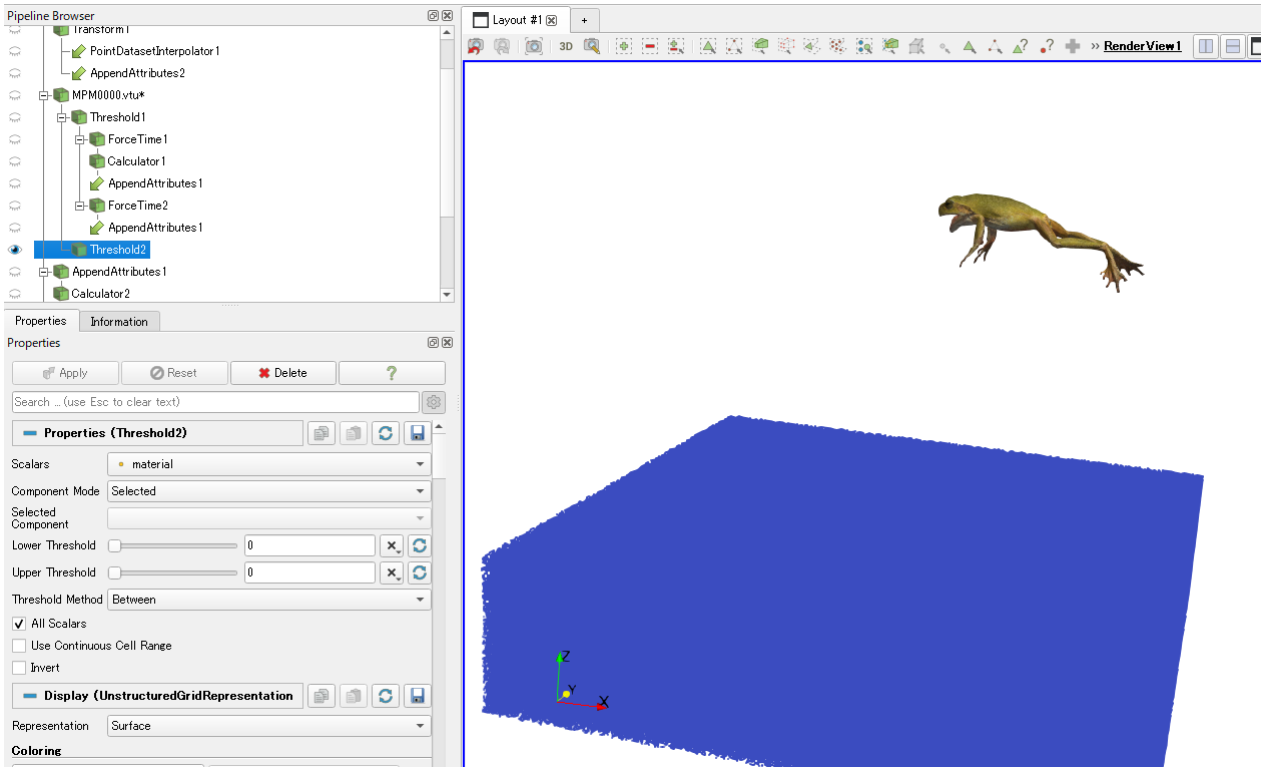
次に，テクスチャを貼るための処理を行う．やらなくていい作用であるはずだが，Texture の TCoords を選択できない現象が起こるので念のため．
TCoords が存在している Transform1（平行移動後の obj 位置情報）と，PointDataSetInterpolator1（obj と初期粒子位置を結合したデータ）を ctl を押しながら同時に選択し，Filters->Alphabetical->Append Attribute を選択し，Apply する．



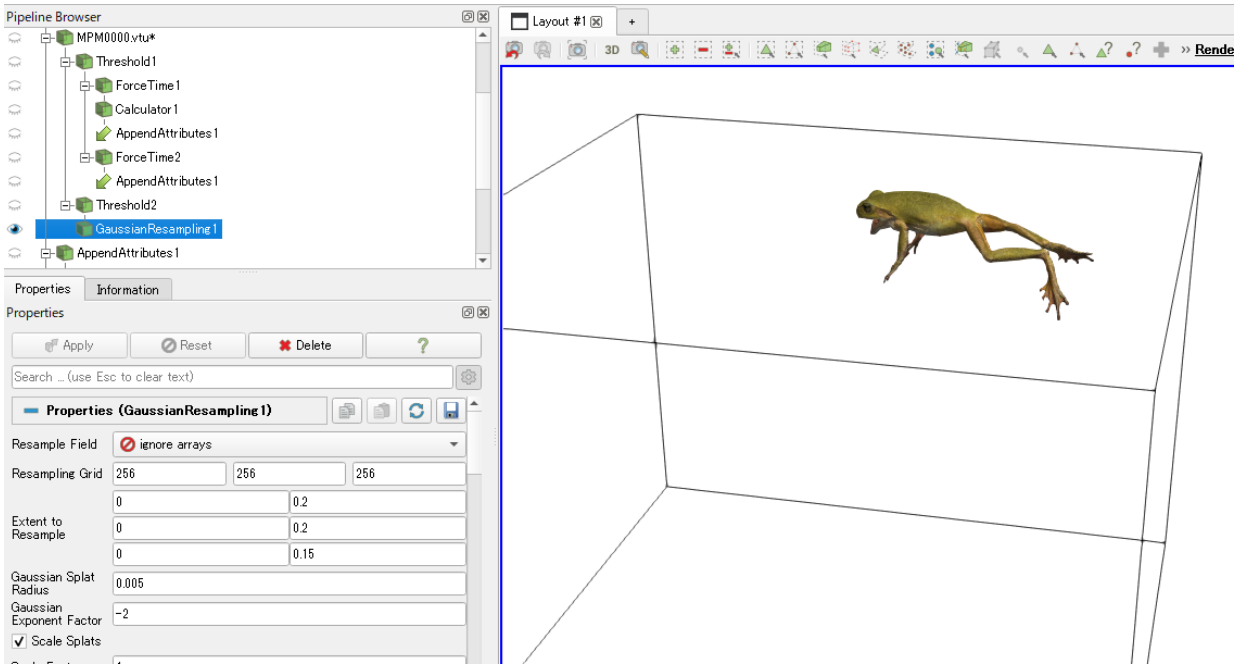
最後に AppendAttributes2 を選択し，Filters->Alphabetical->Warp by Vector を選択し，Vectors に粒子の移動量 U を選択，Texture で元の画像（frog.jpg）を選択して Apply する．色がおかしい場合には，Scalar Coloring を Solid Color にする．



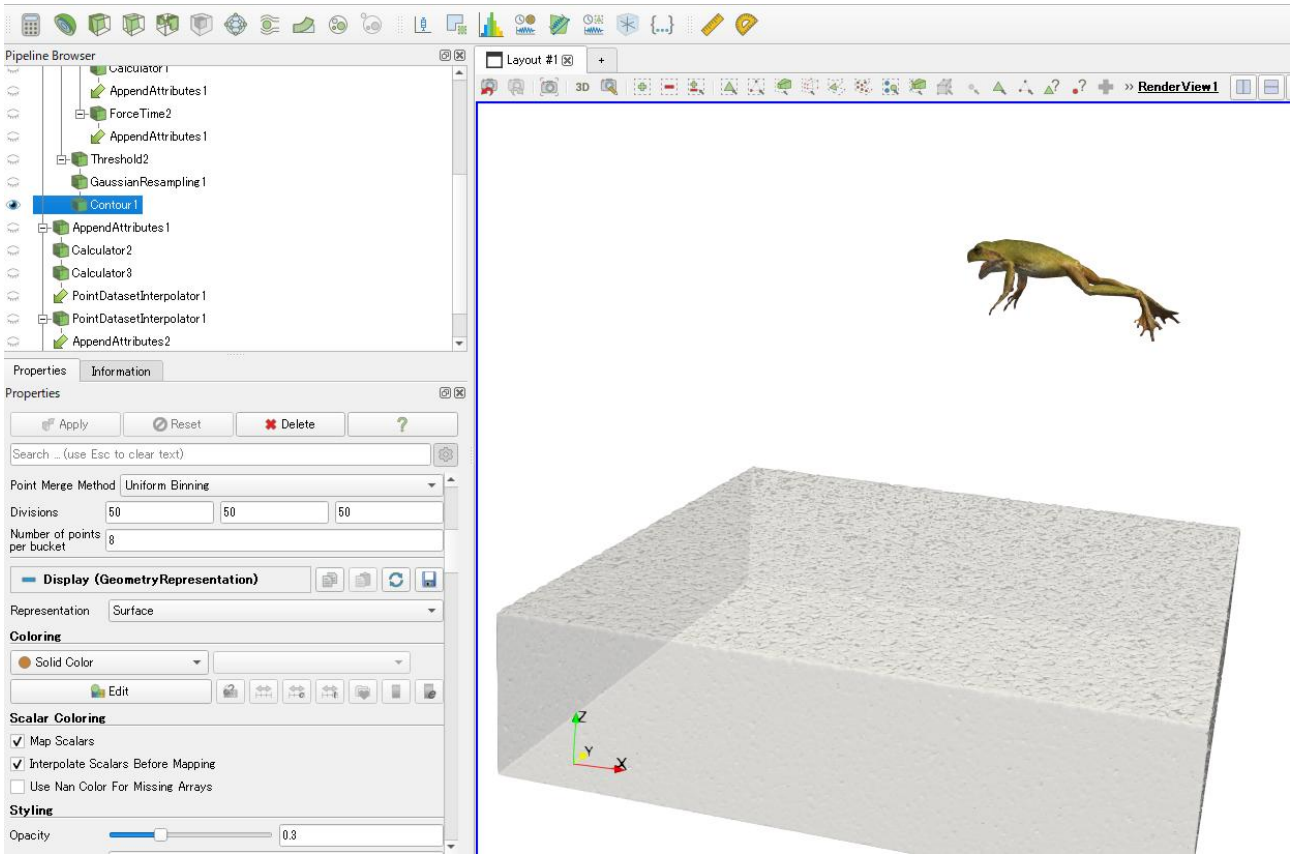
．次に水面を描画する．pipeline から時系列データを選択し，Filters->Alphabetical->Threshold で Lower and Upper Threshold=0 にして（material=0 が水）Apply すると，水面が描画される．



. 同様に，Gaussian Resampling する．

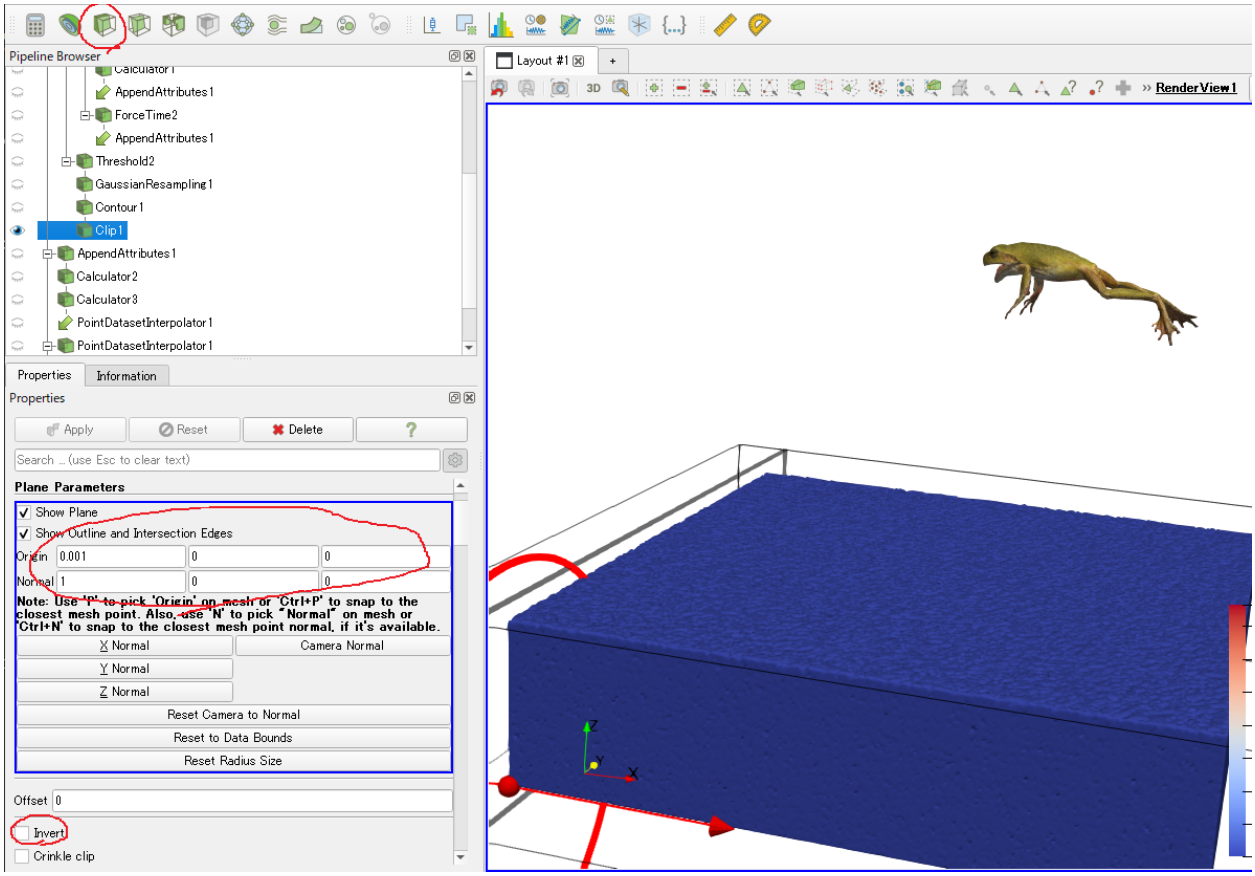


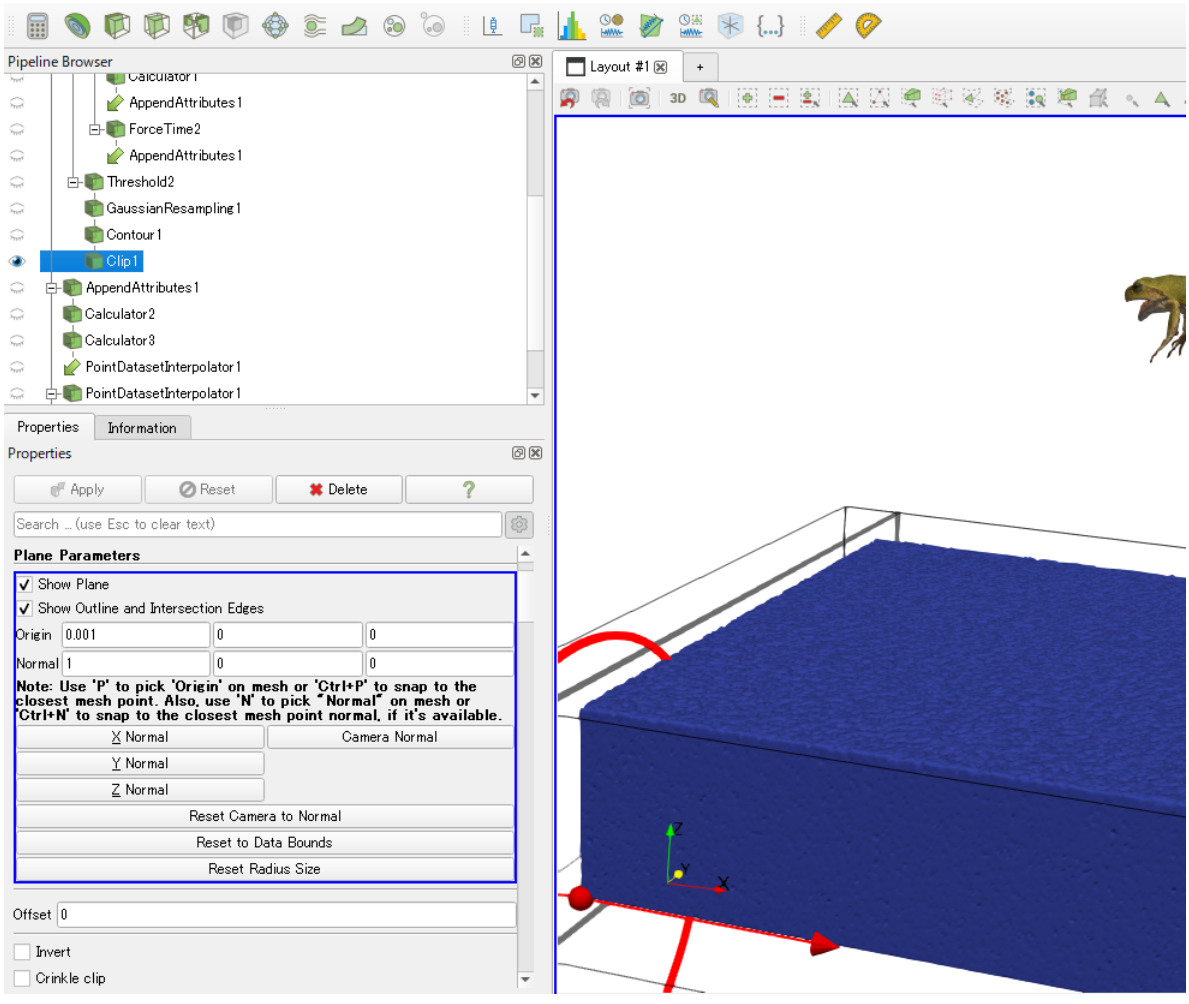
Contour で value range を 0.1 程度にし，Opacity を 0. 3 , Scalar Color を Solid Color にすると図のようになる．



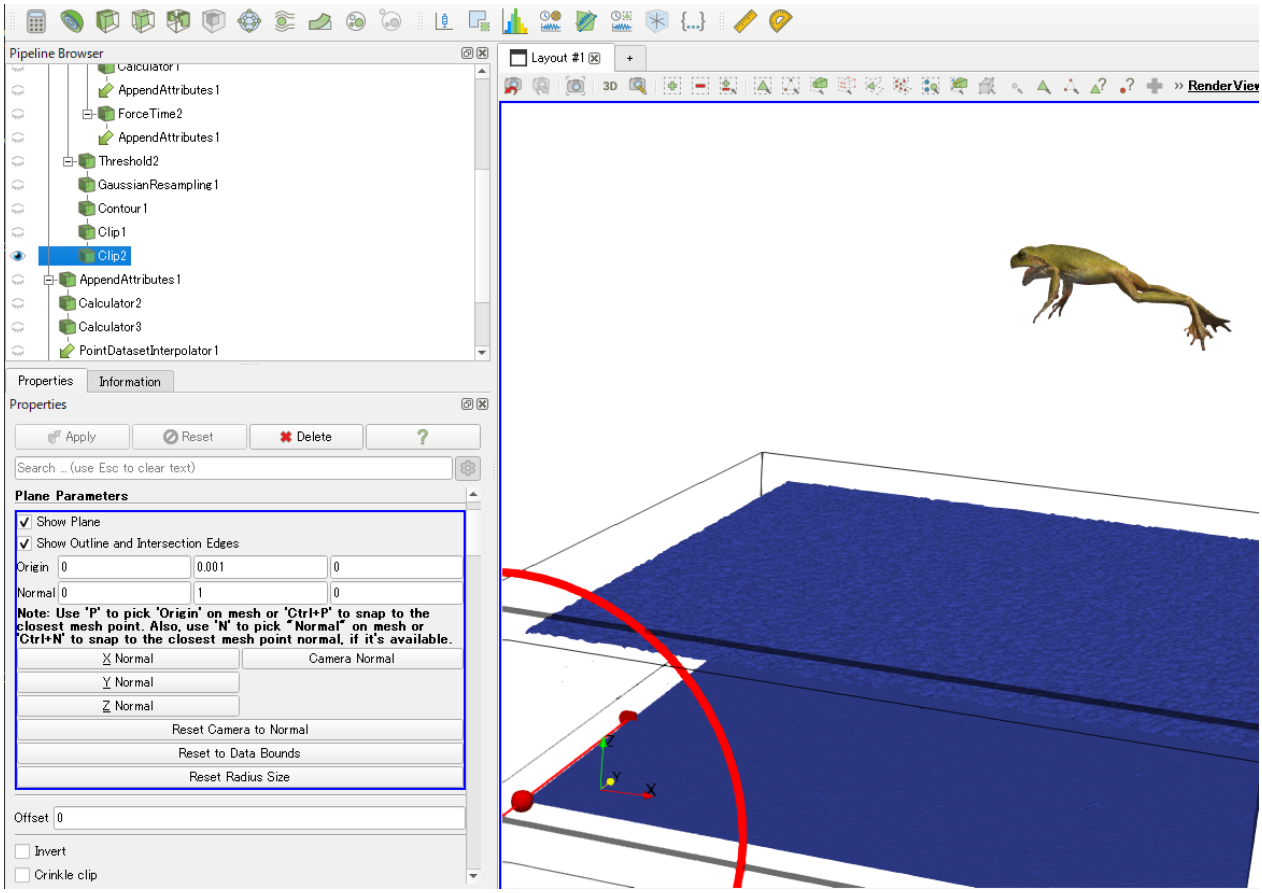
次に，側面と底面がかっこ悪いので削除して水面だけにする．

まず，Clip で x=0 の面を 0.001mm 削る．

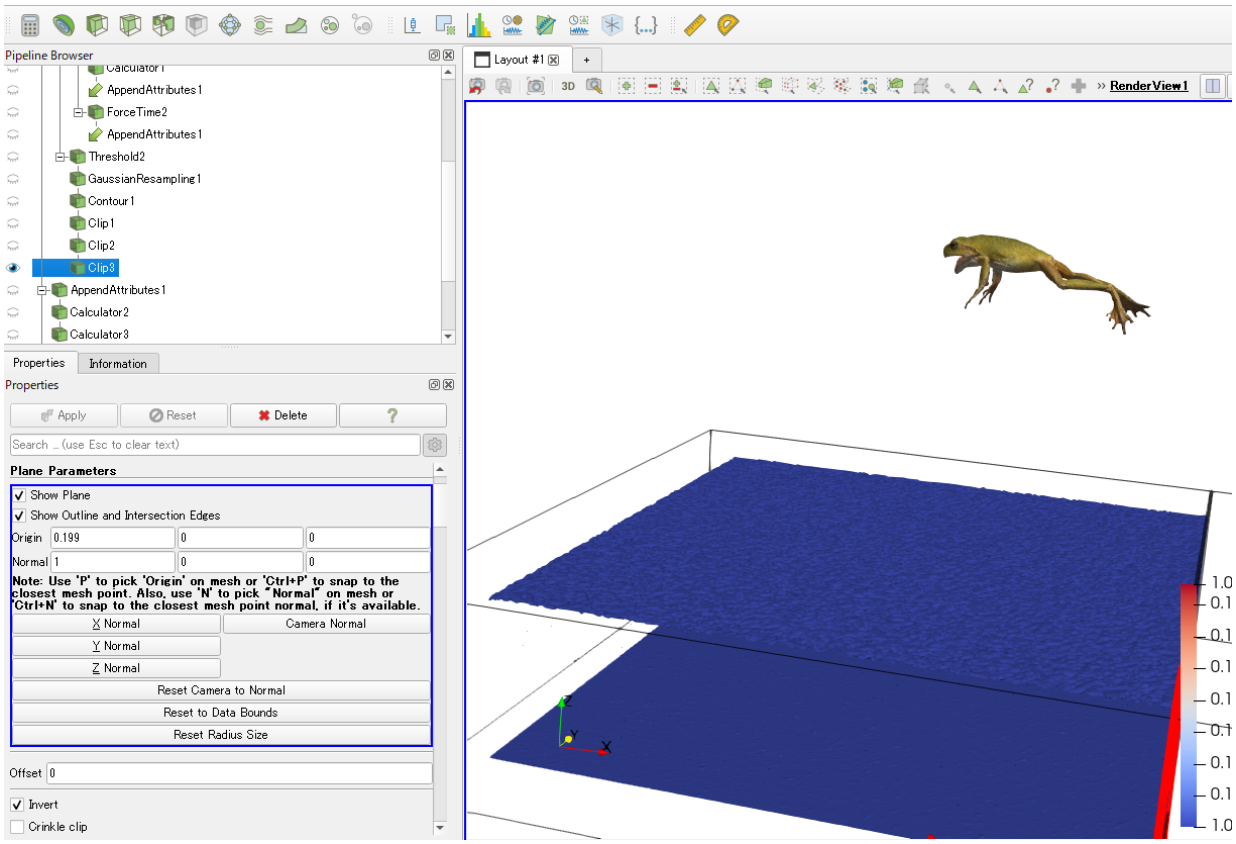

Pipeline Browser
AppendAttributes1
ForceTime2
AppendAttributes1
Threshold2
GaussianResampling1
Contour1
Clip1
AppendAttributes1
Calculator2
Calculator3
PointDatasetInterpolator1
PointDatasetInterpolator1
Properties
Information
Properties
Apply
Reset
Delete
Search ... (use Esc to clear text)
Plane Parameters
Show Plane
Show Outline and Intersection Edges
Origin 0.001 0 0
Normal 1 0 0
Note: Use 'P' to pick 'Origin' on mesh or 'Ctrl+P' to snap to the closest mesh point. Also, use 'N' to pick "Normal" on mesh or 'Ctrl+N' to snap to the closest mesh point normal, if it's available.
X Normal
Camera Normal
Y Normal
Z Normal
Reset Camera to Normal
Reset to Data Bounds
Reset Radius Size
Offset 0
Invert
Crinkle clip
Layout #1
3D
RenderView1


Pipeline Browser
AppendAttributes1
ForceTime2
AppendAttributes1
Threshold2
GaussianResampling1
Contour1
Clip1
AppendAttributes1
Calculator2
Calculator3
PointDatasetInterpolator1
PointDatasetInterpolator1
Properties
Information
Properties
Apply
Reset
Delete
Search ... (use Esc to clear text)
Plane Parameters
Show Plane
Show Outline and Intersection Edges
Origin 0.001 0 0
Normal 1 0 0
Note: Use 'P' to pick 'Origin' on mesh or 'Ctrl+P' to snap to the closest mesh point. Also, use 'N' to pick "Normal" on mesh or 'Ctrl+N' to snap to the closest mesh point normal, if it's available.
X Normal
Camera Normal
Y Normal
Z Normal
Reset Camera to Normal
Reset to Data Bounds
Reset Radius Size
Offset 0
Invert
Crinkle clip
Layout #1
3D

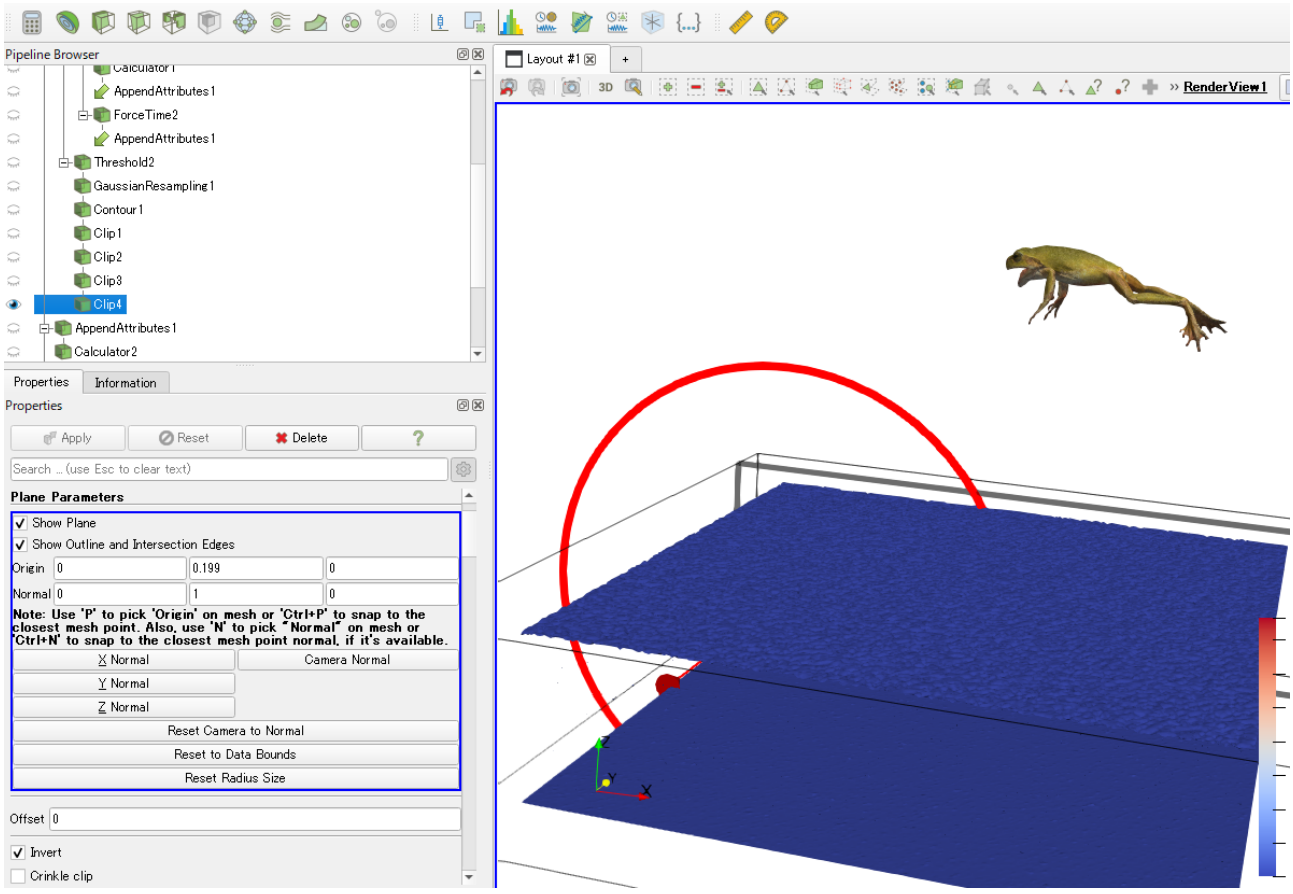
同様に，Clip で y=0 の面を 0.001mm 削る．



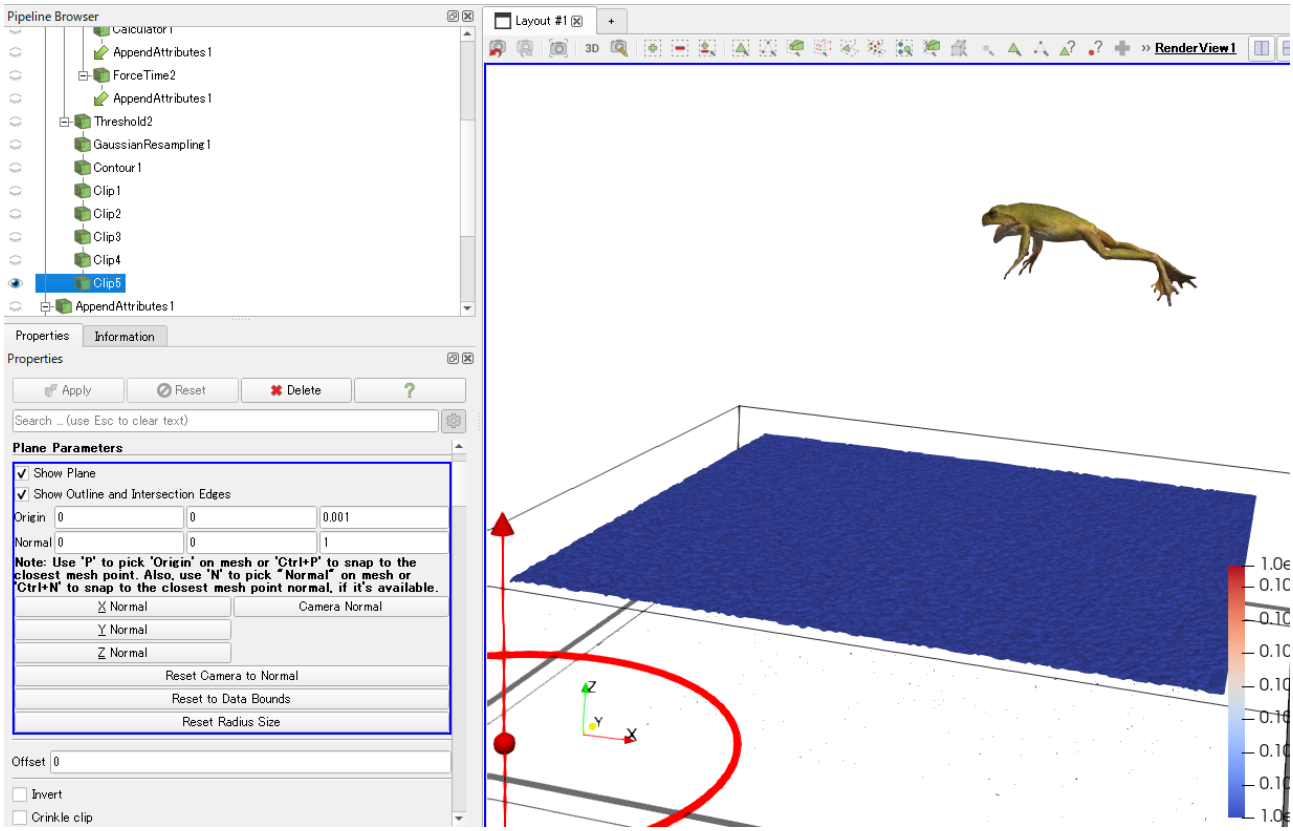
同様に，Clip で x=0.2 の面を 0.001mm 削る．



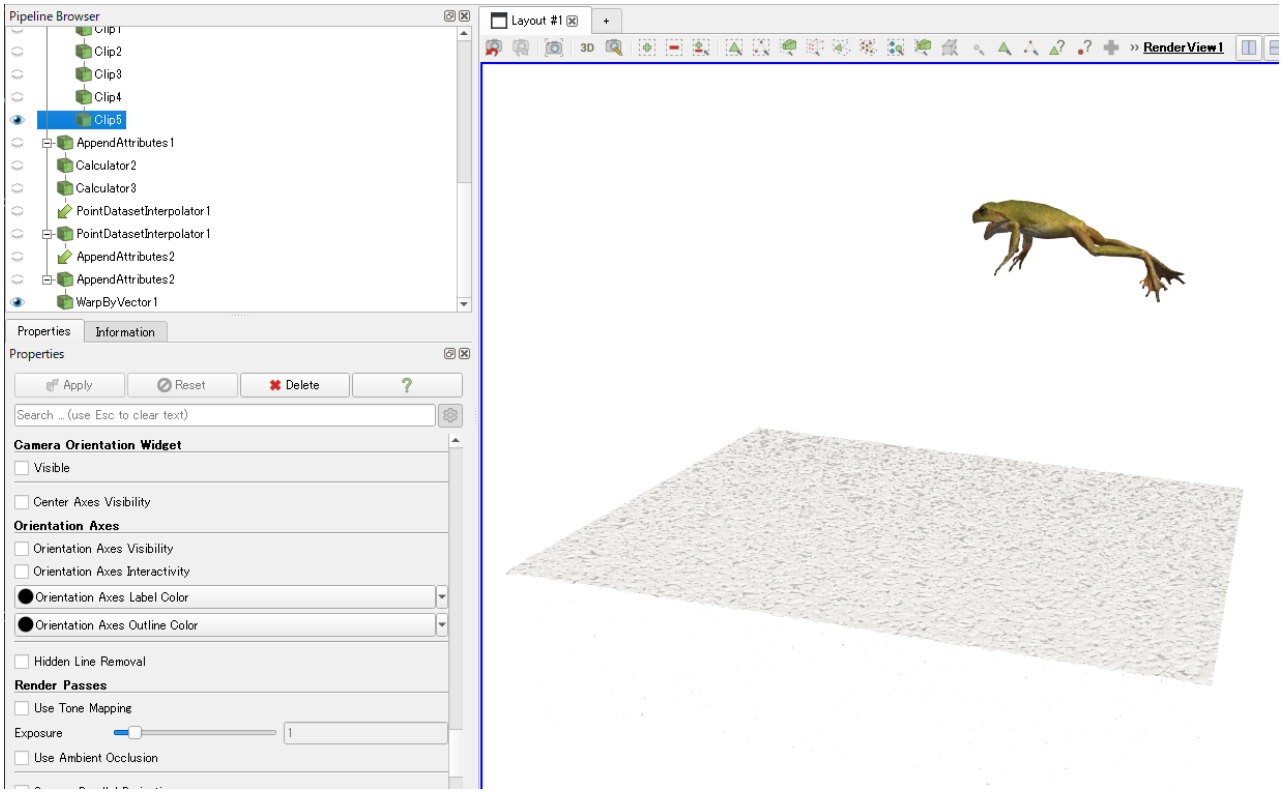
同様に，Clip で y=0.2 の面を 0.001mm 削る．



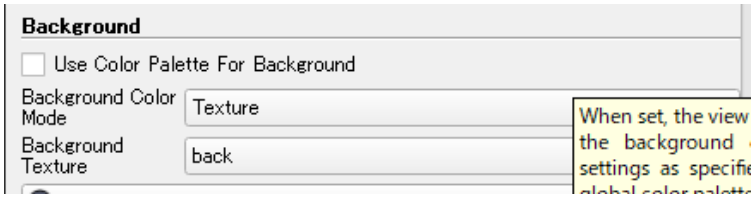
最後に，Clip で z=0 の面を 0.001mm 削る．

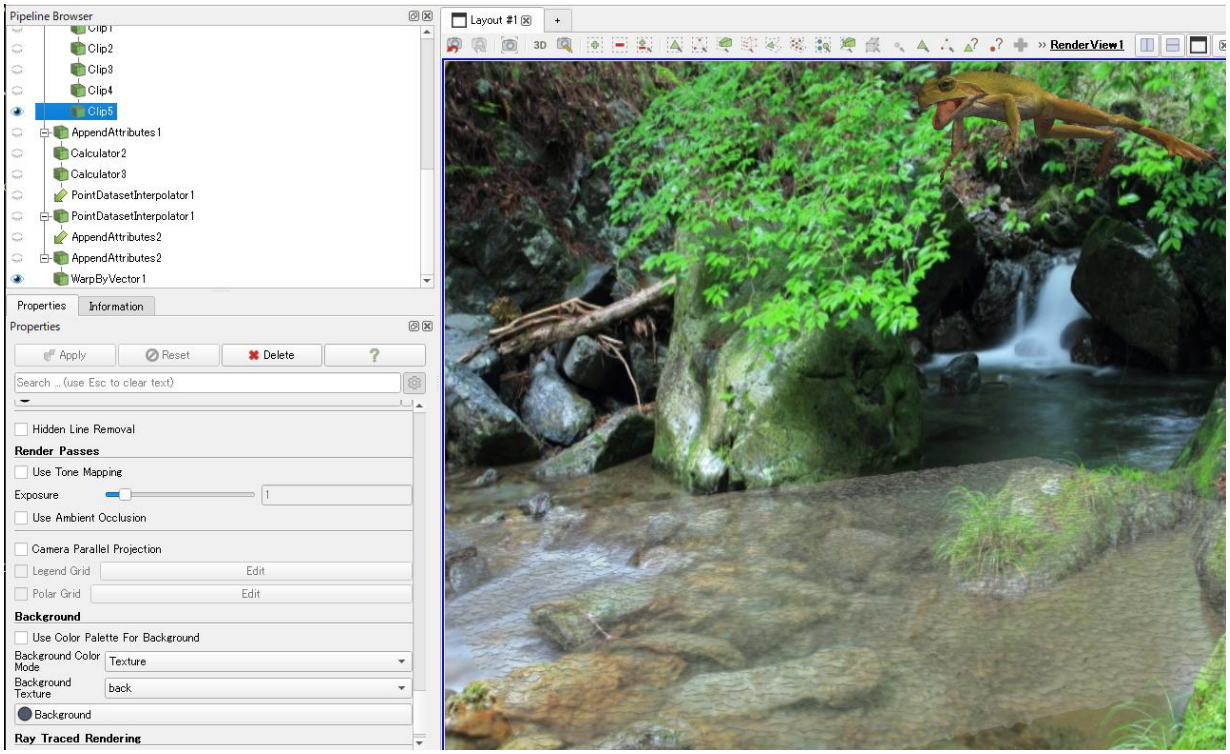


Show plane を外し，水面の色と透明度を調整する．座標系も非表示にしておく．



なんなら，背景も貼れる．適当に back.jpg を貼ってみた．




Pipeline Browser
Clip2
Clip3
Clip4
Clip5
AppendAttributes1
Calculator2
Calculator3
PointDatasetInterpolator1
PointDatasetInterpolator1
AppendAttributes2
AppendAttributes2
WarpByVector1
Properties
Information
Properties
Apply
Reset
Delete
Search ... (use Esc to clear text)
Hidden Line Removal
Render Passes
Use Tone Mapping
Exposure
1
Use Ambient Occlusion
Camera Parallel Projection
Legend Grid
Edit
Polar Grid
Edit
Background
Use Color Palette For Background
Background Color Mode
Texture
Background Texture
back
Background
Ray Traced Rendering
Layout #1
RenderView1