# VTKの基礎

鈴木　将之*

# A Basics of VTK

Masayuki Suzuki*

Visualization Toolkit (VTK)はオープンソースの科学計算向けの可視化とデータ処理ライブラリである。VTK は可視化ソフトウェア ParaView のバックエンドであり、ParaView の主機能は VTK として実装されている。VTK は可視化、データ処理、GUI コンポーネントの提供など多くのポスト処理機能を提供する。本稿では VTK の基本的な使い方として,VTK によるデータ読み込み、主なデータ構造と簡単なフィルタ処理例を述べる。VTK を用いることで、1) ParaView で可視化するためのデータ作成、2) 自ソフトへの様々なデータ形式の読み書きの実装と 3) ParaView 同様の多様なポスト処理機能を、高い再利用性のもと実装することができる。

Keywords: 可視化、ポスト処理、VTK

## 1. はじめに

近年ではオープンソースの科学計算用可視化ソフトウェア ParaView [1]が広く使われている。ParaView は多様な科学計算ファイルの読み込み機能や多くのポスト処理機能を持つ。また、ParaView はクロスプラットフォームかつオープンソースソフトウェアであり、利用可能な環境が多い。

ParaView の主な機能は Visualization Toolkit (VTK) [2]というライブラリで実装されている。ParaView において、各種ファイル読み込み、フィルタやレンダリングなどは VTK が、GUI コンポーネントは Qt [3]が、他機能は各種オープンソースソフトウェアが利用されている。VTK もまた 3 条項 BSD ライセンス [4]が適用されているオープンソースのライブラリであり、ライセンスに従い利用が可能である。

VTK を外部ライブラリとしてソフトウェアに組み込むことで、ParaView の主な機能と同じ機能を実現することが可能となる。ParaView にはマクロやスクリプト機能があり、機能の外部利用が可能な場合もあるが、マニュアルが少ないことと、GUI がない環境では基本的に利用することができない。対し、VTK はマニュアルの整備がなされており主機能の API は安定的であることと、オフスクリーンレンダリングライブラリをリンクすることで、GUI がない環境でもレンダリング以外の機能の利用が可能である。

VTK のユースケースを以下挙げる。

1. ParaView で可視化したい VTK 形式の計算結果ファイル出力を、VTK を用いて実装する。
2. 各種科学計算ファイル読み込みを、VTK を用いて実装する。ソルバ作成者は VTK のデータ構造からソルバのデータ構造への変換のみ実装する。
3. ParaView と同様のポスト処理を VTK で実装し、GUI なしで実行可能にする。実装したポスト処理はソルバに組み込むか、外部ツールとして提供する。
4. 独自の可視化ツールのレンダリング部分を VTK で実装する。

本稿では VTK を用いたファイル読み書き、VTK のデータ構造とフィルタ処理の基本を述べる。VTK の API は継承やコーディング規約により統一的であることから、本稿の内容をおさえておくことで、本稿で触れることができなかったフィル

*アドバンスソフト株式会社　第 2 事業部
2nd Computational Science and Engineering Group, AdvanceSoft Corporation

タなどの利用を容易にすることが期待できる。

本稿では VTK を用いたレンダリングは扱わない。VTK を用いたレンダリングには Qt の基礎知識が必要であることと、環境によっては X などウィンドウシステムが用意されておらずコード例が実行不可であることがあるためである。

VTK は C++で実装されており、C++含めいくつかのプログラミング言語向けの API が提供されている。本稿では環境の構築が容易な Python 版 API を用いる。C++版 API のクラス名、継承関係やメンバ名は Python 版 API と同一なため、C++版 API を利用した場合でも本稿のサンプルコードは十分参考にできる。

サンプルコードの実行に必要な Python 版 VTK のインストール方法例を 2 章に述べる。VTK の機能のうち、ファイル読み書きを 3 章、VTK のデータ構造を 4 章、フィルタ処理の基本を 5 章に紹介する。6 章にまとめを述べる。

## 2. VTK のインストール方法例

本章ではサンプルコードの実行に必要な VTK のインストール方法例を述べる。VTK はオープンソースのライブラリであり、apt、yum や conda-forge など多くのリポジトリからパッケージが提供されている。本稿では最も単純で汎用なインストール方法として pip [5]の例を紹介する。pip によるインストール前に venv などで仮想環境を作ることを強くおすすめする。

Python と pip をインストールした環境において、コード 1 に示すコマンドを実行するとインストールが実行される。コード 1 を実行し正常にインストールされれば、VTK の動的ライブラリ含め、サンプルコードの実行に必要なファイルがインストールされる。

```
pip install vtk
```

コード 1 pip による VTK ライブラリインストールコマンド例

## 3. VTK を用いたファイル読み書き

本章では VTK を用いたファイル読み書きを紹介する。先にファイル読み書きを紹介することで、ParaView 上で既存のファイルの統計やフィルタ適用結果と、出力結果の比較を可能とし、VTK のデータ構造と処理の理解が容易になることを期待する。

コード 2 に VTK の XML 形式構造格子ファイルである vts 形式ファイルの読み込み例を示す。コード 2 は、file_name に指定されたファイルパスを読み込み，読み込み結果を変数 vtk_structured_grid に格納する例である。

```
import vtkmodules.all as vtk

file_name='ファイルパス.vts'
reader = vtk.vtkXMLStructuredGridReader()
reader.SetFileName(file_name)
reader.Update()
vtk_structured_grid = reader.GetOutput()
```

コード 2 vts 形式ファイル読み込み例

コード 3 に VTK の XML 形式非構造格子ファイルである vtu 形式ファイルの読み込み例を示す。コード 3 は、file_name に指定されたファイルパスを読み込み，読み込み結果を変数 vtk_unstructured_grid に格納する例である。コード 2 とコード 3 より、基本的な読み込み処理はファイルによらず同一であり、読み込みクラスを適切に変えることで様々なファイル形式を読み込むことが可能であることが分かる。

```
import vtkmodules.all as vtk

file_name='ファイルパス.vtu'
reader = vtk.vtkXMLUnstructuredGridReader()
reader.SetFileName(file_name)
reader.Update()
vtk_unstructured_grid = reader.GetOutput()
```

コード 3 vtu ファイル読み込み例

VTK における読み込みクラスを表 1 に抜粋する。表 1 に示す通り、VTK では VTK 形式ファイルのほか、いくつか広く使用されている科学計算用ファイルフォーマットをサポートしている。

表 1 読み込みクラス名抜粋

| | |
|---|---|
| vtkXMLImageDataReader | vtkXMLPolyDataReader |
| vtkXMLRectilinearGridReader | vtkSTLReader |
| vtkXMLStructuredDataReader | vtkAVSucdReader |
| vtkXMLUnstructuredDataReader | vtkDICOMImageReader |

コード 4 に VTK の XML 形式構造格子ファイルの一つである vts 形式ファイルの書き込み例を示す。コード 4 において、出力例のために空の構造格子を作成している。コード 4 は、file_name に指定されたファイルパスへ空の構造格子を格納した vts 形式ファイルを書き込む例である。

```
import vtkmodules.all as vtk

# 空の構造格子を生成
vtk_structured_grid = vtk.vtkStructuredGrid()

file_name='tmp.vts'
writer = vtk.vtkXMLStructuredGridWriter()
writer.SetFileName(file_name)
writer.SetInputData(vtk_unstructured_grid)
writer.Write()
```

コード 4 vts 形式ファイル書き込み例

コード 5 に VTK の XML 形式非構造格子ファイルである vtu 形式ファイルの書き込み例を示す。コード 5 において、出力例のために空の非構造格子を作成している。コード 5 は file_name に指定されたファイルパスへ空の非構造格子を格納した vtu ファイルを書き込む例である。コード 4 とコード 5 より、基本的な書き込み処理もまたファイル形式によらず同一であり、書き込みクラスを適切に変えることで、データを様々なファイル形式として書き込むことが可能であることが分かる。

```
import vtkmodules.all as vtk

# 空の非構造格子を生成
vtk_unstructured_grid = vtk.vtkUnstructuredGrid()

file_name='tmp.vtu'
writer = vtk.vtkXMLUnstructuredGridWriter()
writer.SetFileName(file_name)
writer.SetInputData(vtk_unstructured_grid)
writer.Write()
```

コード 5 vtu 形式ファイル書き込み例

## 4. VTK の基本的なデータ構造

本章では VTK のいくつかのデータ構造を紹介する。VTK でよく使用されるデータ構造を以下挙げる。

1. vtkPolyData クラス。ポリゴンなど表面データを格納するためのデータ構造である。
2. vtkImageData クラス。等間隔の構造格子と画像を格納するためのデータ構造である。
3. vtkRectilinearGrid クラス。間隔が不均一の格子状に節点が並んだ構造格子のためのデータ構造である。
4. vtkStructuredGrid クラス。任意座標の節点を持つ構造格子のためのデータ構造である。
5. vtkUnstructuredGrid クラス。非構造格子のためのデータ構造である。

また、上記に格納されるサブのデータ構造として以下がよく使用される。

1. vtkPoints クラス。節点情報を格納する。
2. vtkPointData クラス。節点成分を格納する。
3. vtkCellData クラス。要素成分を格納する。

コード 6 に構造格子を格納するための vtkStructuredGrid クラスの例を示す。コード 6 は 4x3x2 の構造格子を生成し、vts 形式ファイルに出力する例である。コード 6 ではジオメトリのみ生成している。図 1 にコード 6 の出力ファイル可視化結果を示す。

```
import vtkmodules.all as vtk

# 節点情報を生成
points = vtk.vtkPoints()
points.SetNumberOfPoints(24)

point_id = 0
for k in range(2):
    for j in range(3):
        for i in range(4):
            points.SetPoint(
                point_id, i * 1.0, j * 2.0, k * 3.0)
            point_id = point_id + 1

# 構造格子を生成
structured_grid = vtk.vtkStructuredGrid()
# 各軸方向の格子数を設定
structured_grid.SetDimensions(4, 3, 2)
# 節点情報を登録
structured_grid.SetPoints(points)

# 概要を出力
print(structured_grid)

writer = vtk.vtkXMLStructuredGridWriter()
writer.SetFileName("structured_grid.vts")
writer.SetInputData(structured_grid)
writer.Write()
```

コード 6 vtkStructuredGrid クラス使用例

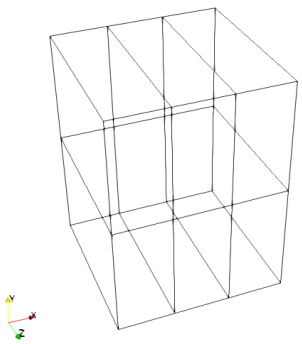

図 1 コード 6 の出力ファイル可視化結果

```
import vtkmodules.all as vtk

# 節点情報を生成
points = vtk.vtkPoints()
points.SetNumberOfPoints(24)

point_id = 0
for k in range(2):
    for j in range(2):
        for i in range(3):
            points.SetPoint(
                point_id, i * 1.0, j * 2.0, k * 3.0)
            point_id = point_id + 1

# 非構造格子を生成
unstructured_grid = vtk.vtkUnstructuredGrid()
# 節点情報を登録
unstructured_grid.SetPoints(points)

# vtkHexahedron 要素を追加
hexa0 = vtk.vtkHexahedron()
pids = [6, 0, 1, 7, 9, 3, 4, 10]
for lid, gid in enumerate(pids):
    hexa0.GetPointIds().SetId(lid, gid)

unstructured_grid.InsertNextCell(
    hexa0.GetCellType(), hexa0.GetPointIds())

hexa1 = vtk.vtkHexahedron()
pids = [7, 1, 2, 8, 10, 4, 5, 11]
for lid, gid in enumerate(pids):
    hexa1.GetPointIds().SetId(lid, gid)

unstructured_grid.InsertNextCell(
    hexa1.GetCellType(), hexa1.GetPointIds())

# この後非構造格子を操作する場合、要素隣接情報などを更新
unstructured_grid.BuildLinks()

# 概要を出力
```

```
print(unstructured_grid)

writer = vtk.vtkXMLUnstructuredGridWriter()
writer.SetFileName("unstructured_grid.vtu")
writer.SetInputData(unstructured_grid)
writer.Write()
```

コード 7 vtkUnstructuredGrid クラス使用例

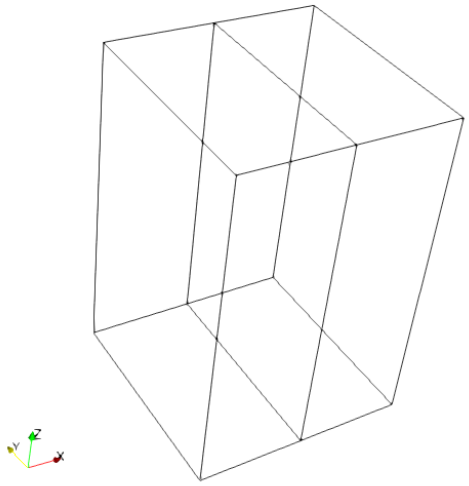

図 2 コード 7 の出力ファイル可視化結果

コード 7 に非構造格子を格納するための vtkUnstructuredGrid クラスの例を示す。コード 7 は 2 個の VTK の六面体要素 vtkHexahedron から構成される非構造格子を生成し、vtu ファイルに出力する例である。コード 7 ではジオメトリのみ生成している。図 2 にコード 7 の出力ファイル可視化結果を示す。

コード 6 とコード 7 では、共通のデータ構造として任意節点座標を格納する vtkPoints クラスを使用している。等間隔の構造格子を格納する vtkImageData クラスなど任意節点座標を使用する必要がないクラスは、vtkPoints クラスを使用しないことに注意いただきたい。

コード 8 に要素成分を格納する例を示す。コード 8 は格子 grid に対し point id という名前の、節点数×1 要素の節点成分を格納するスニペットである。コード 8 に示す通り、VTK において節点成分は vtkPointData クラスのインスタンスに格納する。要素成分もまた、vtkCellData クラスのインスタンスに対し類似した API を呼ぶことで格納することができる。

```
def set_point_scalar(grid):
    # VTK の配列を生成
    point_id_dataarray = vtk.vtkLongLongArray()
    # #tuple * #component の配列を生成.
    point_id_dataarray.SetNumberOfTuples(
        grid.GetNumberOfPoints())
    (point_id_dataarray.
        SetNumberOfComponents(1))
    #  節点 ID を設定
    for i in range(grid.GetNumberOfPoints()):
        point_id_dataarray.SetValue(i, i)
    #  成分名を設定
    point_id_dataarray.SetName("point id")

    # vtkPointData クラスのインスタンスに登録
    grid.GetPointData().AddArray(
        point_id_dataarray)
```

コード 8 節点成分の格納例

## 5. フィルタ処理

VTK と ParaView において、レンダリング以外の処理をフィルタ処理と呼ぶ。断面といった可視化のための処理のほか、補間といったデータ処理や成分の消去といったデータの修正などがフィルタ処理として実装されている。本章ではいくつか基本的なフィルタ処理を紹介する。

コード 9 に ParaView の Threshold フィルタに相当するコード例を示す。コード 9 は、ParaView に含まれるファイル例を読み込み、ある節点成分が 1000 以上の要素のみを抽出し、非構造格子としてファイル出力する例である。図 3 にコード 9 による出力ファイル可視化例を示す。

```
import vtkmodules.all as vtk

# ParaView の Example に含まれる headsq.vti
headsq_path = 'C:/Program Files/ParaView 5.10.1-Windows-Python3.9-msvc2017-AMD64/examples/headsq.vti'
reader = vtk.vtkXMLImageDataReader()
reader.SetFileName(headsq_path)
reader.Update()
headsq = reader.GetOutput()

# ParaView の Threshold フィルタに相当
threshold_filter = vtk.vtkThreshold()
threshold_filter.SetLowerThreshold(1000.0)
threshold_filter.SetInputData(headsq)
threshold_filter.Update()
# threshold_filter の結果は非構造格子として出力される
unstructured_grid = threshold_filter.GetOutput()

writer = vtk.vtkXMLUnstructuredGridWriter()
writer.SetFileName('unstructured_grid.vtu')
writer.SetInputData(unstructured_grid)
writer.Write()
```

コード 9 Threshold クラス使用例

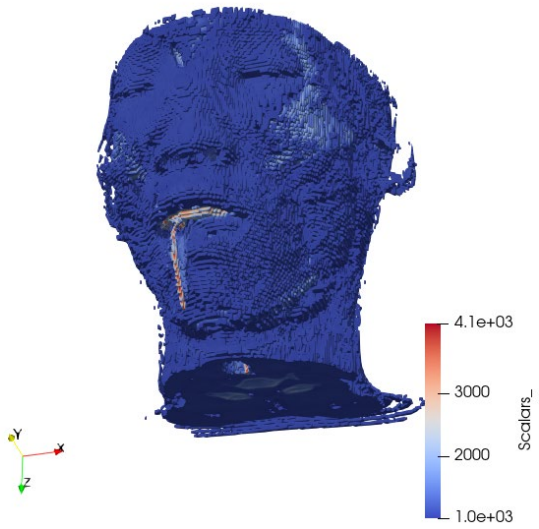


図 3　コード 9 出力ファイル可視化例

コード 10 に ParaView の Clip フィルタに相当するコード例を示す。コード 10 は ParaView に含まれるファイル例を読み込み、vtkClip クラスのデフォルトの XY 断面でクリッピングし、非構造格子として出力する例である。コード 10 ではまた、中間結果を生成することなく読み込み、フィルタと書き込みインスタンスを接続し、最後に一括で更新を行うよう API を呼んでいる。図 4 にコード 10 による出力ファイル可視化例を示す。

```
import vtkmodules.all as vtk

# ParaView の Example に含まれる headsq.vti
headsq_path = 'C:/Program Files/ParaView 5.10.1-Windows-Python3.9-msvc2017-AMD64/examples/headsq.vti'
reader = vtk.vtkXMLImageDataReader()
reader.SetFileName(headsq_path)

# ParaView の Clip フィルタに相当
clip_filter = vtk.vtkClipDataSet()
# フィルタ(読み込みインスタンス)を接続
clip_filter.SetInputConnection(
    reader.GetOutputPort())

writer = vtk.vtkXMLUnstructuredGridWriter()
writer.SetFileName('unstructured_grid.vtu')
# フィルタを接続
writer.SetInputConnection(
    clip_filter.GetOutputPort())
# 接続されている全フィルタを更新
writer.Update()
writer.Write()
```

コード 10 Clip クラス使用例

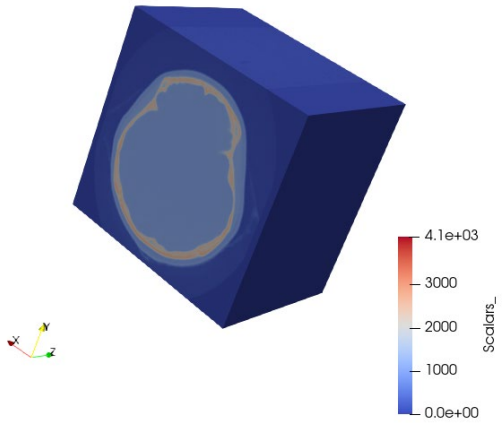


図 4 コード 10 出力ファイル可視化例

コード 9 とコード 10 に示した通り、VTK でフィルタ処理を行うためには概ね以下の順で API を呼び出す。

1. フィルタクラスのインスタンスを生成する。
2. 1 のインスタンスのメンバ関数を呼び出し、パラメータを設定する。
3. 1 のインスタンスの SetInputData(または SetInputConnection) メンバ関数を呼び出し、フィルタの入力を登録する。
4. 1 のインスタンスの Update メンバ関数を呼び出し、接続されているフィルタを全て更新する。
5. 1 のインスタンスの出力を得たい場合、GetOutput メンバ関数を呼び出す。

## 6. まとめ

本稿では VTK の基礎として、基本的なファイル読み書き、データ構造の説明と簡単なフィルタ処理を行った。本稿で示した通り、VTK の API は継承やコーディング規約により統一的であり、非常に可読性が高いことがわかる。すなわち、VTK を用いることで、ポスト処理の実装において高い再利用性と可読性を同時に得ることができる。

本稿では VTK を用いたレンダリングを扱わなかった。VTK はレンダリングのための Qt コンポーネントを提供している。また、VTK は断面の選択、凡例や尺など 3D ビュー上のコンポーネントも提供している。VTK と Qt と組み合わせることで、ポスト処理ソフトウェアにポスト処理のほか、ParaView 同様のレンダリングを容易に実装することが可能となる。

弊社は VTK を用いたソフトウェア開発の実績がある。また VTK の導入支援も行っている。ポスト処理におけるデータ処理やレンダリングでお困りの時はぜひご相談いただきたい。

また、VTK に関する知識を活かして、弊社は ParaView の利用支援サービスを行っている。VTK は ParaView のバックエンドであり、ParaView の新機能やマニュアルにない機能などを、VTK の知識を活かして利用支援を行うことができる。オープンソースの可視化ソフトウェア ParaView をご利用の際には、ぜひ弊社の知見も加えていただきたい。

## 参考文献

[1] ParaView - Open-source, multi-platform data analysis and visualization application. Online. ParaView - Open-source, multi-platform data analysis and visualization application. [n.d.]. Available from: https://www.paraview.org/. [viewed 2023-05-08].

[2] VTK - The Visualization Toolkit. Online. VTK - The Visualization Toolkit. [n.d.]. Available from: https://vtk.org/. [viewed 2023-05-08].

[3] Qt | Cross-platform Software Design and Development Tools. Online. Qt | Tools for Each Stage of Software Development Lifecycle. [n.d.]. Available from: https://www.qt.io/ja-jp/. [viewed 2023-05-08].

[4] VTK/Copyright.txt at master · Kitware/VTK. Online. GitHub. [n.d.]. Available from: https://github.com/Kitware/VTK/blob/master/Copyright.txt. [viewed 2023-05-08].

[5] vtk. Online. PyPI. [n.d.]. Available from: https://pypi.org/project/vtk/. [viewed 2023-05-08].

※ 技術情報誌アドバンスシミュレーションは、アドバンスソフト株式会社 ホームページのシミュレーション図書館から、PDF ファイル（カラー版）がダウンロードできます。（ダウンロードしていただくには、アドバンス/シミュレーションフォーラム会員登録が必要です。）