vtk 文件格式

2023 年 12 月 12 日

可视化工具包 (vtk) 提供了许多源和编写器对象对流行的数据文件格式进行读写。vtk 也提供一些其自己的文件格式。创建另一种文件格式的目的在于为多种数据集提供一致的数据表示方案以及简单的软件间数据通信方式。当可能时，我们建议使用更广泛使用的文件格式。相反，这里描述的 vtk 格式可以代替使用。注意其他工具可能不支持这些格式。

vtk 中有两种不同的文件格式。最简单的是传统的串行格式，其易于以手动或程序方式读写。

simple legacy formats-简单传统格式

传统 vtk 格式包含五个部分。

1. 第一个部分是文件的版本和标识符。这一部分包含单行:' vtk DataFile version x.x'. 除版本号 x.x 外，这一行必须如所示完全一致，而版本数随 vtk 发行版变化而变化 (注：当前版本数为 3.0，版本 1.0 和版本 2.0 与版本 3.0 文件兼容)
2. 第二个部分是标题。标题由以结束符" 结尾的字符串组成。其最大字符数为 256。它可用于描述数据和其他任何相关信息。
3. 下一部分是文件格式。该文件格式描述文件的类型是 ASCII 或二进制文件。单词 ASCII 或 BINARY 必须出现在这一行。
4. 第四部分为数据结构。几何部分描述了数据集的几何与拓扑。这一部分包含以 DATASET 开头的关键字的以及其后描述数据集类型的关键字的单行组成。随后，根据数据集的类型，其他关键字/数据组合定义了真实数据。
5. 最后一部分描述数据集的属性。这一部分以关键字 POINT_DATA 或者 CELL_DATA 开始，后跟一个整数分别表示点或单元的数量。POINT_DATA 或 CELL_DATA 谁先出现不重要。其他关键字/数据组合定义真实数据集的属性值 (如标量，向量，张量，发现，纹理坐标或场量)。

文件格式概述如图 1 所示。前三部分是强制性的，但其他两个是可选的。因此你可以通过操作系统文件操作或 vtk 过滤器来合并数据，灵活地混合和匹配数据集属性以及几何图形。关键字不区分大小写，且可通过空格分隔。

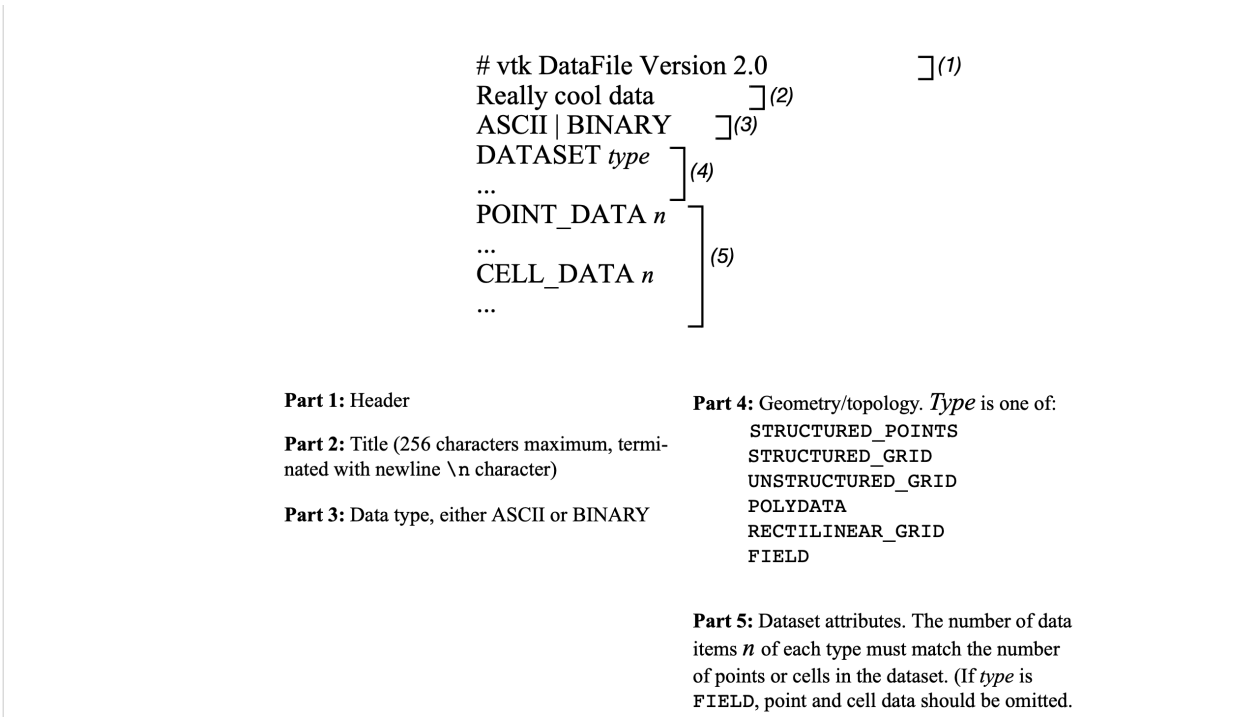

Figure 1 Overview of five parts of VTK data file format.

在描述数据文件格式前请注意一下几点。

- dataType 是 bit,unsigned_char,char,unsigned_short,unsigned_int,int,unsigned_long,long, float 或者 double 其中之一。这些关键字用于描述数据的形式，既为了从文件中读取数据也为构建合适的内部数据对象。并非所有类都支持所有的数据类型。
- 无论是 ASCII 文件还是二进制文件，所有的关键字短语都是以 ASCII 形式编写。文件的二进制部分 (如果在二进制部分) 是是数据正确的，即定义点坐标、标量、单元索引等的数字 (是正确的)。
- 索引是 0 偏移的，即第一个点 id 是 0.
- 如果数据属性和几何/拓扑部分同时在一个文件中，那么在数据属性部分中的数值数量必须与几何/拓扑部分中定义的点或单元数量相等。
- 单元类型或索引是 int 类型。
- 二进制数据必须紧接前一个 ASCII 关键字和参数序列中的换行 () 字符之后放入文件中。
- 几何拓扑描述必须出现在数据属性描述之前。

二进制文件　只要满足两个条件，二进制文件就可以在不同电脑系统之间移植。第一保证数据的字节序正确；其次确保每个数据类型的长度一致。

大部分时候，vtk 为你管理二进制文件的字节序。当你在一个电脑中写一个二进制文件，并从另外一个电脑中读取，表示数据的字节必要时将会自动转换。例如，在 Sun 架构中二进制文件以大端序写入，而在 PC(译者注：应该是 x86 架构中) 是以小端序储存的。这就导致在 Sun 工作站写入的文件需要在 PC 中读入时进行字节转换 (见类 vtkByteSwap 的实现细节)。此处描述的 vtk 数据文件是以大端序形式写入的。

然而，一些文件格式没有显式地定义字节序形式。你将会发现根据初始系统的不同，外部程序或者类 vtkVolume16Reader,vtkMCubesReader 以及 vtkMCubesWriter 读写的数据可能会有不同的字节顺序。在这种情况下，vtk 允许你使用以下方法指定字节序:

```
SetDataByteOrderToBigEndian()
SetDataByteOrderToLittleEndian()
```

二进制文件的另外一个问题是操作系统可能使用不同字节数表示整数或其他原生类型。例如，某些 64 位操作系统可能 8 个字节表示一个整数，而其他操作系统使用 4 个字节。目前 vtk 无法在不兼容数据长度的操作系统间转移二进制文件。在这种情况下，请改用 ASCII 文件格式。

数据集格式 可视化工具包 (vtk) 支持五种不同的数据集格式: 结构化点，结构化网格，直线网格，非结构化网格和多边形数据。具有隐式拓扑的数据 (结构化数据，如 vtkImageData 和 vtkStructuredGrid) 按 x 增加最快的顺序排列，然后是 y，然后是 z。这些格式如下。

- 结构化点
  该文件格式支持一维，二维和三维的结构化点数据集。其 dimensions 值 n_x, n_y, n_z 必须大于等于 1. 其 spacing(间距) 值必须大于 0.(注：在版本 1.0 的数据文件中，spacing(间距) 被称为 “长宽比”，ASPECT_RATIO 在版本 2.0 的数据结构中仍可使用，但不建议)

```
DATASET STRUCTURE_POINTS
DIMENSIONS nx ny nz
ORIGIN x y z
SPACING sx sy sz
```

- 结构化网格
  该文件格式支持一维，二维和三维的结构化网格数据集。其 dimensions 值 n_x, n_y, n_z 必须大于等于 1. 点坐标在 POINTS 节中的数据定义。其包含了每点的 x-y-z 数值。

```
DATASET STRUCTURE_GRID
DIMENSIONS nx ny nz
POINTS n dataType
p0x p0y p0z
p1x p1y p1z
```

```
...
p(n-1)x p(n-1)y p(n-1)z
```

- 直线型网格
  直线型网格定义了具有规则拓扑和沿 x-y-z 坐标轴对齐的半规则拓扑数据集。几何由三个单调递增的坐标值列表定义，每个列表分别代表 x-y-z 坐标轴。其拓扑由指定的网格维度定义，其必须大于等于 1.

```
DATASET RECTILINEAR_GRID
DIMENSIONS nx ny nz
X_COORDINATES nx dataType
x0 x1 ... x(nx-1)
Y_COORDINATES ny dataType
y0 y1 ... y(ny-1)
Z_COORDINATES nz dataType
z0 z1 ... z(nz-1)
```

- 多边形数据
  多边形数据集由曲面图形图元顶点 (和多边形顶点), 直线 (和折线)，多边形 (各种类型) 和三角形的任意组合组成。多边形数据由 POINTS，VERTICES,LINES, POLYGONS 或者 TRIANGLE_STRIPS 部分定义。POINTS 的定义与我们看到的结构化网格的定义相同。VERTICES, LINES，POLYGONS 或者 TRIANGLE_STRIPS 关键词定义了多边形数据集拓扑。每一个这样的关键词需要两个参数：单元数 n 和单元列表的大小 size。单元列表大小是表示列表所需的所有整数值。(例如，每个单元的 numPoints 之和与连通性指数) VERTICES,LINES,POLYGONS 或者 TRIANGLE_STRIPS 不是必须的。

```
DATASET POLYDATA
POINTS n dataType
p0x p0y p0z
p1x p1y p1z
...
p(n-1)x p(n-1)y p(n-1)z

VERTICES n size
numPoints0,i0,j0,k0,...
numPoints1,i1,j1,k1,...
...
numPointsn-1,in-1,jn-1,kn-1,...
```

```
LINES n size
numPoints0,i0,j0,k0,...
numPoints1,i1,j1,k1,...
...
numPointsn-1,in-1,jn-1,kn-1,...

POLYGONS n size
numPoints0,i0,j0,k0,...
numPoints1,i1,j1,k1,...
...
numPointsn-1,in-1,jn-1,kn-1,...

TRIANGLE_STRIPS n size
numPoints0,i0,j0,k0,...
numPoints1,i1,j1,k1,...
...
numPointsn-1,in-1,jn-1,kn-1,...
```

- 非结构化网格
  非结构化网格数据集包含任意可能的单元类型的组合。非结构化网格由点，单元和单元类型定义。CELLS 关键字要求两个参数：单元数 n 和单元列表的大小 size。单元列表大小是表示列表所需的所有整数值 (例如，每个单元的 numPoints 之和与连通性指数)。CELL_TYPE 关键字要求一个参数：单元数 n。这个值应与 CELLS 关键词指定的值相匹配。单元类型数据是每个单元一个指定单元类型的整数值 (见 vtkCell.h 或图二)

```
DATASET UNSTRUCTURED_GRID
POINTS n dataType
p0x p0y p0z
p1x p1y p1z
...
p(n-1)x p(n-1)y p(n-1)Z

CELLS n size
numPoints0,i,j,k,l,...
numPoints1,i,j,k,l,...
numPoints2,i,j,k,l,...
...
numPointsn-1,i,j,k,l,...

CELL_TYPE n
```

```
type0
type1
type2
...
typen-1
```

- 场
  场数据是没有拓扑和几何结构的通用数据格式，也没有特定维度。通常，场数据和数据集的点或单元相关联。然而，如果指定 FIELD 类型为数据集类型 (见图一)，那么就定义了通用 vtk 数据对象。使用下节介绍的格式定义场。也可见在 158 页的 “Working With Field Data” 和在 7 页的 “Examples” 章节的第四个例子。

数据集属性格式 可视化工具包 (vtk) 支持下面五种数据集属性：标量 (一到四个分量)，向量，法线，纹理坐标 (一维，二维和三维)，3X3 张量和场量。此外，还可以定义使用与标量数据相关联的 RGBA 颜色规范的查找表。点和单元都支持数据集属性。

每种属性数据的类型都有一个与之相关的数据名 (dataName)。这是一个用于标识具体数据的字符串 (没有嵌入空格)。数据名 (dataName) 被 vtk 读取器用来提取数据。因此一个文件中可以包含多个相同类型的属性数据。例如定义在数据集点中的两个不同的标量场，压力和温度，可以包含在同一个文件中。(如果在 vtk 读取器中没有指定合适的数据名 (dataName)，那么只从文件中提取第一个该类型的数据。)

- 标量
  标量定义包含查询表的规范。查询表的定义是可选的。如果没有指定，将使用默认的 vtk 表 (tableName 应为”default”)。同时也注意 numComp 参数也是可选的–默认分量数量为 1.(参数 numComp 的范围必须在 (1,4) 之间 (含)；在 vtk2.3 之前的 vtk 版本不支持这个参数)

```
SCALARS dataName dataType numComp
LOOKUP_TABLE tableName
s0
s1
...
sn-1
```

颜色标量 (即直接映射到颜色的 unsigned char 值) 的定义根据每个标量值 (nValues) 的数量而变化。如果文件格式为 ASCII，那么颜色标量使用 (0,1) 之间的标量值 (nValues) 个浮点值定义。如果文件格式是二进制文件，那么数据流由每个标量值的 nValues 个 unsigned char 值组成。

```
COLOR_SCALARS dataName nValues
c00 c01 ... c0(nValues-1)
```

```
c10 c11 ... c1(nValues-1)
...
c(n-1)0 c(n-1)1 ... c(n-1)(nValues-1)
```

- 查询表
  用于标识查询表的 tableName 字段是一个字符串 (没有插入空格)。vtk 读取器使用这个标签提取特定的表格。
  查找表中的每一个条目都是一个 rgba[4](红-绿-蓝-alpha) 数组 (alpha 是不透明度, 其中 alpha=0s 是透明的)。如果文件格式是 ASCII，那么查询表中的值是 (0,1) 之间的浮点数。如果文件格式是二进制文件，数据流必须是每个表格条目由四个 unsigned char 值组成的。

```
LOOKUP_TABLE tableName size
r0 g0 b0 a0
r1 g1 b1 a1
...
rsize-1 gsize-1 bsize-1 asize-1
```

- 向量

```
VECTORS dataName dataType
v0x v0y v0z
v1x v1y v1z
...
v(n-1)x v(n-1)y v(n-1)z
```

- 法线
  假设法线已标准化 |n|=1.

```
NORMALS dataName dataType
n0x n0y n0z
n1x n1y n1z
...
n(n-1)x n(n-1)y n(n-1)z
```

- 纹理坐标
  支持一维，二维和三维的纹理坐标。

```
TEXTURE_COORDINATES dataName dim dataType
t00 t01 ... t0(dim-1)
t10 t11 ... t1(dim-1)
...
t(n-1)0 t(n-1)1 ... t(n-1)(dim-1)
```

- 张量
  当前仅支持 3X3 实值对称张量。

```
TENSORS dataName dataType
t^0_00 t^0_01 t^0_02
t^0_10 t^0_11 t^0_12
t^0_20 t^0_21 t^0_22

t^1_00 t^1_01 t^1_02
t^1_10 t^1_11 t^1_12
t^1_20 t^1_21 t^1_22

...
t^(n-1)_00 t^(n-1)_01 t^(n-1)_02
t^(n-1)_10 t^(n-1)_11 t^(n-1)_12
t^(n-1)_20 t^(n-1)_21 t^(n-1)_22
```

- 场数据
  场数据本质上是数据数组的数组。定义场数据意味着给场命名且指定了其包含数组的数量。那么对于每一个数组定义了数组名 arrayName(i), 数组分量数量 numComponents, 数组中元组数量 numTuples 以及数据类型 dataType。

```
FIELD dataName numArrays
arrayName0 numComponents numTuples dataType
f00 f01 ... f0(numComponents-1)
f10 f11 ... f1(numComponents-1)
...
f(numTuples-1)0 f(numTuples-1)1 ... f(numTuples-1)(numComponents-1)

arrayName1 numComponents numTuples dataType
f00 f01 ... f0(numComponents-1)
f10 f11 ... f1(numComponents-1)
...
```

```
f(numTuples-1)0 f(numTuples-1)1 ... f(numTuples-1)(numComponents-1)

...
arrayName(numArrays-1) numComponents numTuples dataType
f00 f01 ... f0(numComponents-1)
f10 f11 ... f1(numComponents-1)
...
f(numTuples-1)0 f(numTuples-1)1 ... f(numTuples-1)(numComponents-1)
```

示例 第一个示例是由多边形面表示的立方体。我们在六个面定义了一个单分量标量，法线和场量。标量数据和八个顶点相关。定义了一个和点标量相关的八色查询表。

```
# vtk DataFile Version 2.0
Cube example
ASCII
DATASET POLYDATA
POINTS 8 float
0.0 0.0 0.0
1.0 0.0 0.0
1.0 1.0 0.0
0.0 1.0 0.0
0.0 0.0 1.0
1.0 0.0 1.0
1.0 1.0 1.0
0.0 1.0 1.0
POLYGONS 6 30
4 0 1 2 3
4 4 5 6 7
4 0 1 5 4
4 2 3 7 6
4 0 4 7 3
4 1 2 6 5

CELL_DATA 6
SCALARS cell_scalars int 1
LOOKUP_TABLE default
0
1
2
```

```
3
4
5
NORMALS cell_normals float
0 0 -1
0 0 1
0 -1 0
0 1 0
-1 0 0
1 0 0
FIELD FieldData 2
cellIds 1 6 int
0 1 2 3 4 5
faceAttributes 2 6 float
0.0 1.0 1.0 2.0 2.0 3.0 3.0 4.0 4.0 5.0 5.0 6.0

POINT_DATA 8
SCALARS sample_scalars float 1
LOOKUP_TABLE my_table
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
LOOKUP_TABLE my_table 8
0.0 0.0 0.0 1.0
1.0 0.0 0.0 1.0
0.0 1.0 0.0 1.0
1.0 1.0 0.0 1.0
0.0 0.0 1.0 1.0
1.0 0.0 1.0 1.0
0.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0
```

下一个示例是尺寸为 3X4X5 的体积。因为没有定义查询表，用户要么自己在 vtk 中定义，要么使用默认查询表。

```
    # vtk DataFile Version 3.0
Volume example
ASCII
DATASET STRUCTURED_POINTS
DIMENSIONS 3 4 6
ORIGIN 0 0 0
SPACING 1 1 1
POINT_DATA 72
LOOKUP_TABLE default
0 0 0 0 0 0 0 0 0 0 0 0
0 5 10 15 20 25 25 20 15 10 5 0
0 10 20 30 40 50 50 40 30 20 10 0
0 10 20 30 40 50 50 40 30 20 10 0
0 5 10 15 20 25 25 20 15 10 5 0
0 0 0 0 0 0 0 0 0 0 0 0
```

第三个示例是一个非结构化网格，其包含 19 种 vtk 网格中的 12 种。其中包含向量和标量数据。

```
    # vtk DataFile Version 3.0
Unstructured Grid Example
ASCII

DATASET UNSTRUCTURED_GRID
POINTS 27 float
0 0 0   1 0 0   2 0 0   0 1 0   1 1 0   2 1 0
0 0 1   1 0 1   2 0 1   0 1 1   1 1 1   2 1 1
0 1 2   1 1 2   2 1 2   0 1 3   1 1 3   2 1 3
0 1 4   1 1 4   2 1 4   0 1 5   1 1 5   2 1 5
0 1 6   1 1 6   2 1 6

CELLS 11 60
8 0 1 4 3 6 7 10 9
8 1 2 5 4 7 8 11 10
4 6 10 9 12
4 5 11 10 14
6 15 16 17 14 13 12
6 18 15 19 16 20 17
4 22 23 20 19
3 21 22 18
3 22 19 18
```

```
2 26 25
1 24

CELL_TYPES 11
12
12
10
10
7
6
9
5
5
3
1

POINT_DATA 27
SCAlARS scalars float 1
LOOKUP_TABLE default
0.0 1.0 2.0 3.0 4.0 5.0
6.0 7.0 8.0 9.0 10.0 11.0
12.0 13.0 14.0 15.0 16.0 17.0
18.0 19.0 20.0 21.0 22.0 23.0
24.0 25.0 26.0
VECTORS vectors float
1 0 0   1 1 0   0 2 0   1 0 0   1 1 0   0 2 0
1 0 0   1 1 0   0 2 0   1 0 0   1 1 0   0 2 0
0 0 1   0 0 1   0 0 1   0 0 1   0 0 1   0 0 1
0 0 1   0 0 1   0 0 1   0 0 1   0 0 1   0 0 1
0 0 1   0 0 1   0 0 1
```

第四个即最后一个示例表示为场的数据。你可能希望查看 158 页"Working With Field Data"，以了解如何操作该数据 (下面的数据场可以在 $\$VTK\_DATA_{R}OOT/Data/financial.vtk$)

```
# vtk DataFile Version 2.0
Financial data in vtk field formats
ASCII
FIELD financialData 6
TIME_LATE 1 3188 float
29.14 0.00 0.00 11.71 0.00 0.00 0.00 0.00
...(more stuff - 3177 total values)...
```