自由离散元后处理软件（ÅpenPost）设计说明（V1.0）

目录

一、概要

1.1 开发背景

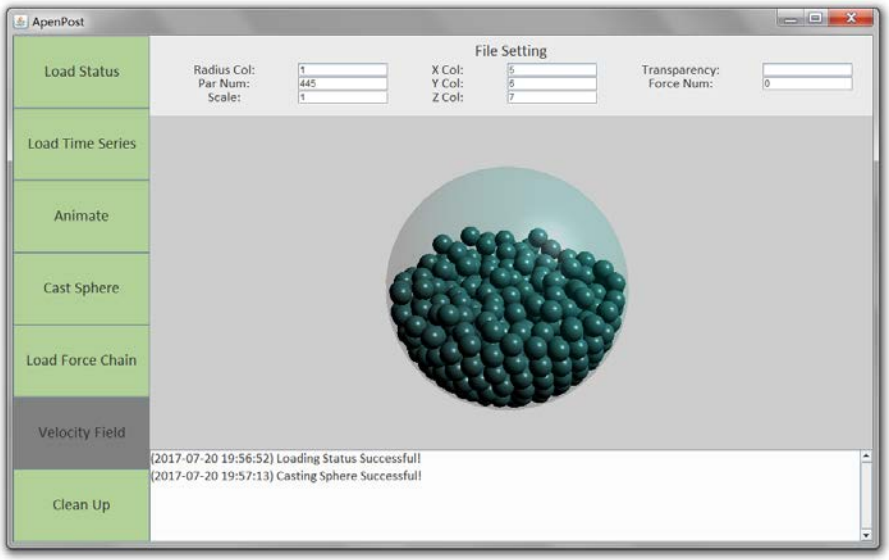

图 1-1 ApenPost 软件界面示意图

颗粒状物质是工业生产生活中经常出现的原材料形态，具有强非线性等重要的特点。为了直观模拟颗粒物质的运动，并加强对颗粒物质力学理论的理解，随着计算机科学的发展，离散元方法（Discrete Element Method, DEM）是近三十年发展起来的可以用来描绘颗粒流动的数值模拟仿真方法。

本软件 ÅpenPost 是基于复旦大学黄骏副教授开发的自由离散元数值计算软件 ÅpenDEM 所开发的相应后处理软件，软件的编写基于高度面向对象的 Java 编程语言，融合了 Java 3D 和 Swing 等主要接口，主要可用于读取离散元方法结果文件，并能够将计算结果进行可视化展示及对其进行后处理操作。软件的主体设计由黄骏教授与我（姚凯）共同完成，编写和维护主要由我独立完成。软件的设计、开发与维护时间由 2017 年 1 月至 2017 年 7 月历时半年，具备了离散元方法基本的后处理功能，实现了对离散元方法结果文件的三维可视化。

ÅpenPost 后处理软件使用了 Java 语言中用于开发三维图形界面的 Java 3D 应用程序接口，同时为了追求美观及实用的人机交互界面，选用 Java 中的 Swing 接口对用户交互界面进行开发，使软件的操作更为简洁明了。本软件主要功能包括但不限于以下几项：读取并加载颗粒群的时刻状态、读取并加载颗粒群的时间序列状态流动画、进行力链网络图的绘制、进行时刻速度场云图的绘制，以及进行时间序列速度场云图动画的绘制等。

1.2 运行环境

由于 Java 语言的强移植性，本软件可以在 Windows、Linux、Mac 等多个平台环境中使用，并可以通过 Eclipse 等多种 Java 语言编译器对源代码进行编译。本软件所使用到的 Java 3D 等外接接口需要用户自行从网络上下载。

1.3 主类及子类的说明

源代码中 class MainClass 下为主类 MainClass 的全部代码，class GBC 下为子类 GBC 的全部代码。子类 GBC 针对 GridBagLayout 界面布局，设置子类 GBC 的目的是为了便于各 Swing 对象的统一排版。编译时将主类和子类置于同一项目中并直接编译主类即可运行该程序。

二、主要接口

2.1 Java 3D 接口

2.1.1 Java 3D 介绍

Java 3D API 是 Sun 公司定义的用于实现计算机三维图形显示的一个应用程序接口（API, Application Programming Interface），它的出现为 Java 在 3D 领域的强有力的涉足提供了无限可能。Java 3D 作为一门程序面向对象语言 Java 的衍生子包，继承了 Java 语言的所有优良特性，如摒弃和优化了许多 C 语言中不易理解的概念（如指针操作和内存管理）以及具有强大的平台移植性（可便利地移植于 Windows、Linux、Mac OS、IOS、Android 等多种平台）。它可以灵活适用于各种三维程序和软件的编写，适用范围相当之广。

2.1.2 主程序 Java 3D 基本结构

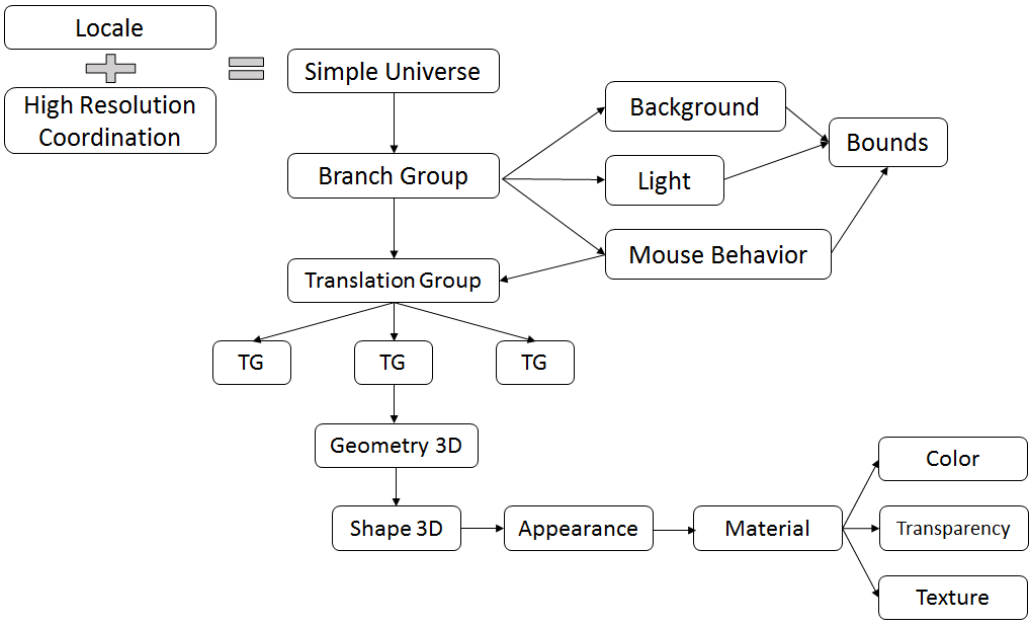

图 2-1 ÅpenPost 主程序 Java 3D 基本结构图

绝大部分 Java 3D 程序均由一个虚拟世界 Simple Universe 构成。一个 Simple Universe 由 Java 3D 内置的 Locale 场所类和 High Resolution Coordination 高精度坐标类所组成，可以在电脑中显示极高精度的坐标。一个 Simple Universe 父类中包含了用户所想要在 3D 画布中进行操作的所有子对象类。Branch Group 内部存放该三维世界 Simple Universe 中的世界背景 Background 类，声音 Sound 类，光源 Light 类以及鼠标行为 Mouse Behavior 类等等，而 Bounds 类作为以上几类的子节点规定了这些类的作用范围，即超出此 Bounds 类对象 bound 范围的区域不受 Branch Group 子节点背景、声音、光源及鼠标行为等的影响。

Branch Group 的唯一子类是 Translation Group，可以存放一个几何体组，该几何体组总是同时受到同一个作用于该 Translation Group 的坐标变换作用。正因如此，常常在 Translation Group 上建立多个子 Translation Group 节点，用以对不

同的几何体组进行不同的类型不同程度的坐标变换。本软件中亦使用 Java 3D 的该特点，通过建立不同的 Translation Group 节点对不同几何体组进行相应操作。

每一个 Translation Group 均可以存放多个 Geometry 3D 子类节点，即可存放多个几何形体。每个 Geometry 3D 节点由多个 Shape 3D 子节点构成，可存放如 Sphere，Cylinder 等多种内置或用户自行编写的 Shape 3D 子节点。而所有对于 Shape 3D 几何形体的外观的描述则存放于 Appearance 类中。Appearance 类是 Shape 3D 类的子节点，又是 Material 类的父节点。一个 Appearance 对象对应一个 Material 对象。一个 Material 对象中可以存放多种材料属性，如 Color 对象、Transparency 对象或 Texture 对象等。

以上即为一个 Java 3D 程序所包含的节点架构。

2.2 Swing 接口

2.2.1 Swing 介绍

由 Java 语言编写开发而成的程序或软件，下至小型游戏上至大型工程软件均离不开强大的 Swing 人机交互模块。Swing 模块是一个用于开发 Java 应用程序用户交互界面的开发工具包，具有十分强大的与用户进行人机交互；即从用户处读取信息以及将信息反馈给用户的功能。Swing 接口以抽象窗口工具包（AWT, Abstract Window Toolkit）为根基，与用户的本地计算机图形界面进行交互，使跨平台的 Java 应用程序可以灵活使用任意插拔的外观风格。Java 的开发人员利用 Swing 接口丰富的功能和模块化的组件来创建优雅美观的人机交互界面。Swing 接口提供的功能类举例如下：软件布局设定、按键（单选、多选等）、文本输入输出框、菜单、事件监听器（鼠标、键盘行为监听器）等等。在一个 Java Application 或 Java Applet 的总窗体 Frame 中，Frame 作为 Swing 接口中最大的容器，可以装下各种 Swing 子组件作为其子节点，其中就包括如上所述的按键、框体、菜单、事件监听器等等。

2.2.2 ÅpenPost 的 Swing 界面布局

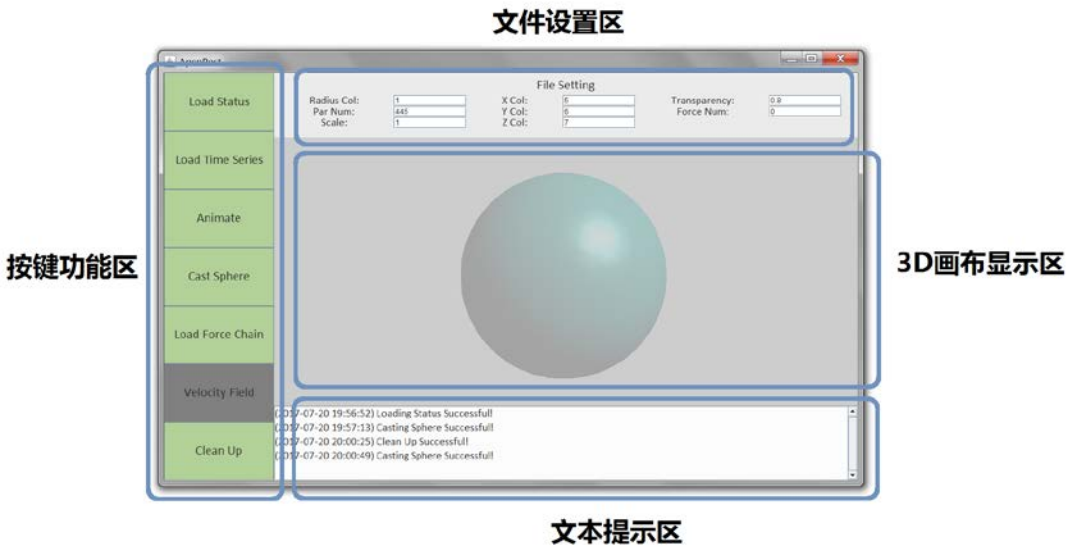

图 2-2 ÅpenPost 后处理软件 Swing 界面结构

如图 2-2 所示，ÅpenPost 后处理软件主界面主要分为四个主要区块，分别为按键功能区、文本提示区、文件设置区以及最为重要的 3D 画布显示区。按键功能区放置的是数个可用于进行三维图像显示或三维动画显示等的 Swing 接口的按键类对象，其所使用的类主要为 JButton 类，属于私有变量。文本提示区的功能类似于 ABAQUS 或 ANSYS 等有限元和有限差分力学软件，可向用户警示某特定操作行为是否合法、成功并对用户进行必要的提示等。

文件设置区可供用户设置所读取的离散元方法计算结果文件格式，即用户可自行定义离散元方法程序所得结果文本文件中例如颗粒半径、三维坐标等值所在的列数，并可对放大系数、透明细数、颗粒数目、力链链对数目等等进行定义。此外，用户未来还将可以通过 3D/2D 的切换框对离散元后处理文件的三维和二维显示进行即时切换，该功能目前正在进行进一步的设计与开发。

三、程序功能

3.1 程序功能总览

Åpen Post 后处理软件主要功能包括但不限于以下几项：读取并加载颗粒群的时刻状态、读取并加载颗粒群的时间序列状态流动画、进行力链网络图的绘制、进行时刻速度场云图的绘制，以及进行时间序列速度场云图动画的绘制等。在本节中，笔者将就这些主要功能进行关于开发原理的详细阐述。

软件其余功能还包括 Cast Sphere 功能、Velocity Field 开关功能及 Clean Up 功能等。对于 Cast Sphere 功能，使用者可使用 Cast Sphere 按键进行球体的“投掷”，即在三维画布中生成一个自定坐标及半径，并可自定透明度的球体，对于 Velocity Field 开关功能，当该按键置灰时加载的文件或文件流均不考虑速度场云图，而当该按键置绿时加载的文件或文件流均直接考虑速度场云图。对于 Clean Up 功能，在使用者按下 Clean Up 功能按键后该程序将会直接清空三维画布中的所有对象。

对于时刻状态加载功能及时刻速度场云图功能，该软件引入了文件加载功能，即在按下相关功能按键键之后弹出文件单选加载框，即读取某一帧的颗粒群状态；对于时间序列状态流动画加载功能及时间序列速度场云图动画功能，该软件引入了文件流加载功能，即在按下相关功能按键键之后弹出文件复选加载框，即读取某自定数量帧的颗粒群的状态流。

3.2 状态加载

3.2.1 时刻状态加载（Load Status）

离散元方法后处理软件最基本的功能之一为根据结果文本文件生成颗粒群并显示每一个颗粒的三维位置。为加载时刻状态，首先程序会读取一个文本文件，并根据用户所指定的颗粒半径生成颗粒。由于 Java 3D 默认三维形体 Shape 3D 对象生成于坐标原点处，需根据用户所指定的颗粒坐标所在列数将每个颗粒依次进行坐标变换平移到其正确坐标位置处。为此该软件引入了必要的文件加载功能，即在按下相关功能按键键之后软件会弹出文件单选加载框，随之读取某一帧的颗粒群状态。

以 x 轴坐标为例，记 $x_{i,j}$ (i表示第i个颗粒，j表示第j帧) 为第i个颗粒在第j帧时刻的x轴坐标。另记总颗粒数为n_{par}，则可获得向量

$$X_j = [x_{1,j} \quad x_{2,j} \quad x_{3,j} \quad \cdots\cdots \quad x_{n_{par},j}]^T \tag{3-1}$$

同理

$$Y_j = [y_{1,j} \quad y_{2,j} \quad y_{3,j} \quad \cdots\cdots \quad y_{n_{par},j}]^T \tag{3-2}$$

$$Z_j = [z_{1,j} \quad z_{2,j} \quad z_{3,j} \quad \cdots\cdots \quad z_{n_{par},j}]^T \tag{3-3}$$

在读取结果文件后，程序会将各颗粒在该帧的位置坐标信息分别存入以上 3 个向量中，并根据以上 3 个向量去对各颗粒进行平移变换。通过生成 Shape 3D 中的球体对象，并对每一个球体对象作简单平移即可将所有球形颗粒平移至其坐标位置处并显示在 3D 画布区域中。如图 3-1 所示：

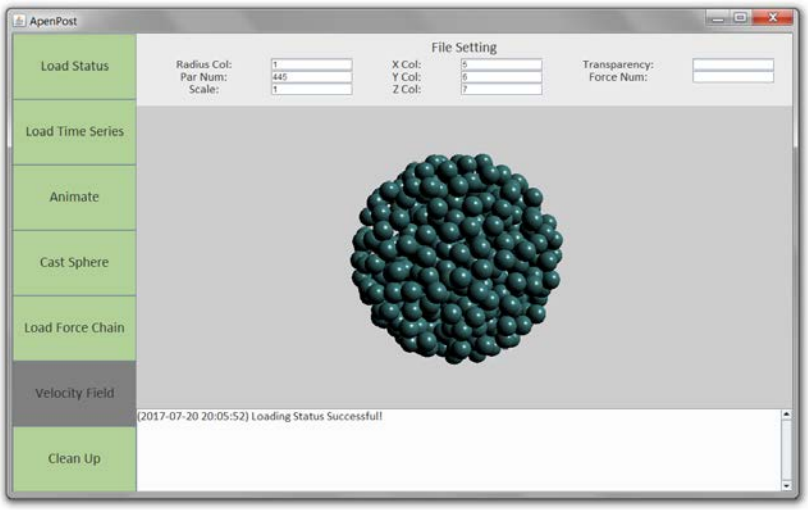

图 3-1 对颗粒群的时刻状态加载示意图

3.2.2 时间序列状态流动画加载（Load Time Series + Animate）

时间序列状态流指的是指定数目个时刻状态的集合。若需加载时间序列状态流动画，则需要同时加载指定数目个时刻状态。为此该软件引入了必要的文件流加载功能，即在按下相关功能按键键之后弹出文件复选加载框，即读取某自定数量帧的颗粒群的状态流，如图 3-2 所示：

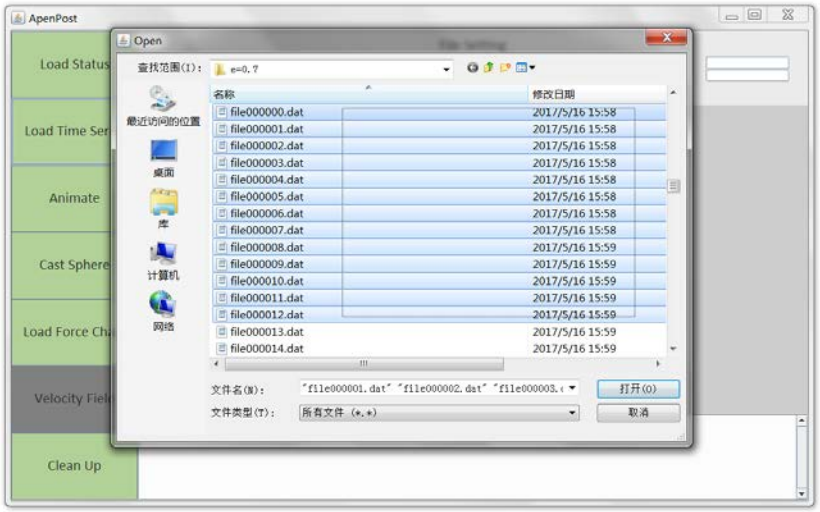

图 3-2 文件复选加载框示意图

在程序实现中，记总帧数为n_{frame}，则可以获得由n_{frame}个上述向量所构成的坐标张量X，

$$X = [X_1 \quad X_2 \quad X_3 \quad \cdots\cdots \quad X_{n_{frame}}] \tag{3-4}$$

其中

$$X_1 = [x_{1,1} \quad x_{2,1} \quad x_{3,1} \quad \cdots\cdots \quad x_{n_{par},1}]^T$$

$$X_2 = [x_{1,2} \quad x_{2,2} \quad x_{3,2} \quad \cdots\cdots \quad x_{n_{par},2}]^T$$

$$\cdots\cdots \tag{3-5}$$

$$X_i = [x_{1,i} \quad x_{2,i} \quad x_{3,i} \quad \cdots\cdots \quad x_{n_{par},i}]^T$$

$$\cdots\cdots$$

$$X_{n_{frame}} = [x_{1,n_{frame}} \quad x_{2,n_{frame}} \quad x_{3,n_{frame}} \quad \cdots\cdots \quad x_{n_{par},n_{frame}}]^T$$

从第j帧到第$j+1$帧的过渡显示中，需要将每一个颗粒进行不同程度的平移变换，其中以x方向为例，第i个颗粒需要在x方向上平移$(x_{i,j+1}-x_{i,j})$的距离。三维空间中的另外两个方向与此类同。在读取结果文件后，程序会将各颗粒在各帧的位置坐标信息分别存入以上张量中。在将每一个颗粒进行n_{frame}次平移变换后即可得到时间序列状态流动画。显然，在总帧数n_{frame}越大的情况下，即每一次颗粒群的平均平移位移越小的情况下，动画显示的光滑度和平滑度越高。

此外，在通过 Load Time Series 按键加载时间序列文件完毕后，用户需点击 Animate（动画生成）按键以生成动画。

3.3 力链网络图（Force Chain）

在颗粒力学方法中常常使用力链网格图来表示每对颗粒之间存在的力的大小。本软件设计力链网络图加载功能，通过下述原理加载三维力链网络图。

单击 Load Force Chain 按键将打开文件框，允许用户选择单一力链网格文件进行力链网格图的呈现。文件格式要求如图 4-6 所示：

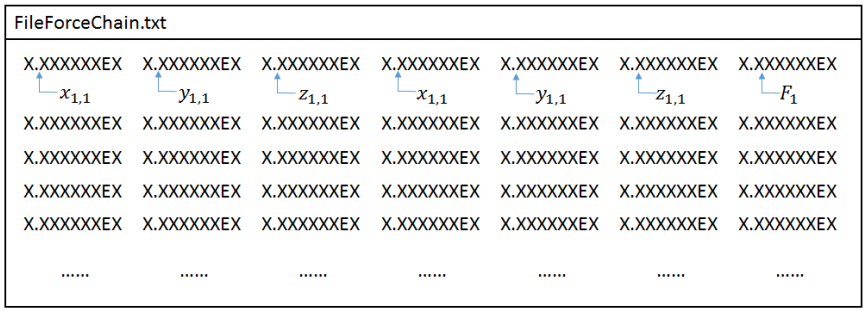

图 3-3 力链网络图加载文本文件格式要求

输入文件格式要求如上图所示，第i行中第 1 列至第 3 列为第i个力链所对应的 2 个颗粒中的第 1 个颗粒的 3 个坐标值$x_{1,i}, y_{1,i}$,和 $z_{1,i}$，第i行中第 4 列至第 6 列为第i个力链所对应的 2 个颗粒中的第 1 个颗粒的 3 个坐标值$x_{2,i}, y_{2,i}$,和 $z_{2,i}$，第i行中第 7 列为该力对所对应力值的大小F_i。

在建立力链时，为平滑棱角并增加美观度，每个力链采用“药丸型圆柱”三维几何形体表示，即除建立一联结两点的圆柱体之外，在两点处分别建立两个球体，三者合为一形成药丸形状，如下示意图所示。为方便起见，将每个力链所对应的药丸型圆柱看作一个 TransformGroup，如此为之则在循环文件时仅需要每循环一行建立一个 TransformGroup 即可。

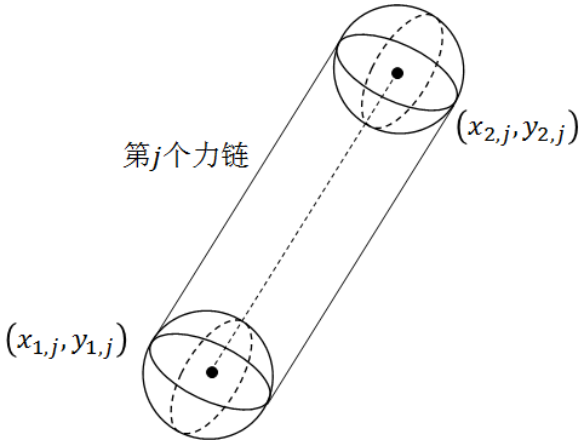

图 3-4 “药丸型圆柱”的力链三维形体示意图

总力链网络图显示效果如下所示：

图 3-5 总力链网络图显示效果示意图

在代码实现中，首先需要规定函数输入变量，即自变量：

```
public TransformGroup myCylinder(float x1, float y1, float z1, float x2, float y2, float z2, float radius,
int xclarity, int yclarity, int sphclarity, Appearance ap)
```

其中 x1，y1，z1 代表点 1 的坐标；x2，y2，z2 代表点 2 的坐标；radius 代表圆柱及球的共同半径，xclarity 及 yclarity 为圆柱在切向和法向方向上的 2 个清晰度参数，具体表示在该圆柱体所在局部坐标系下横轴和竖轴方向的分点数，这 2 个值越大，则三维圆柱体越精确。类似地，sphclarity 代表在球体所在局部坐标系下横轴和竖轴方向的共同分点数。ap 则为圆柱体和两个球体三者共同使用的 Appearance 外观类对象。

为建立圆柱体，首先需要通过输入变量进行必要的数学计算。通过两点之间距离公式可以求得圆柱体的高度，并在画布原点处（Java 3D 一般默认在画布原点处建立 Shape3D 对象）以力值为半径，以两点间距为高建立圆柱体对象：

```
float height = (float) Math.sqrt(Math.pow(x1 - x2, 2) + Math.pow(y1 - y2, 2) + Math.pow(z1 - z2, 2));
Cylinder cylin = new Cylinder(Math.abs(radius), height, 1, xclarity, yclarity, ap);
```

由于需要将某一圆柱体从原点处作坐标变换平移并三维旋转至该圆柱体所对应力链的两个端点：点 1 和点 2 处，此处使用到四元数概念并通过四元数方法将原圆柱体进行旋转。四元数是数学家 William Rowan Hamilton 公爵于 1843 年提出的数学和图形学概念，是其一生中最为伟大的发现之一。四元数有时亦称作四元组或四元数组，由于其简便性，四元数在计算数学领域及计算机图形学领域中有着相当重要的应用。

一个四元数可以看成是一个标量和一个三维向量的组合。如果用符号实部w来表示标量，虚部V来表示向量。则可以将一个四元数q写成以下形式：

$$q = [w, V] \tag{3-6}$$

亦可以写成如下形式：

$$q = [w, (x, y, z)] \tag{3-7}$$

简单而言，一个四元数包含四个标量数值，而后三个数值实际上是一个向量的三个相互独立的分量。因此，正规化四元数可以表示为

$$q = [q_0, q_1, q_2, q_3] \tag{3-8}$$

其模量即为

$$|q| = {q_0}^2 + {q_1}^2 + {q_2}^2 + {q_3}^2 \tag{3-9}$$

在三维程序中，四元数可用于表示任何绕某固定轴的旋转，如下公式所示：

$$\begin{cases} q_0 = \cos\left(\dfrac{\alpha}{2}\right) \\ q_1 = \sin\left(\dfrac{\alpha}{2}\right)\cos(\beta_x) \\ q_2 = \sin\left(\dfrac{\alpha}{2}\right)\cos(\beta_y) \\ q_3 = \sin\left(\dfrac{\alpha}{2}\right)\cos(\beta_z) \end{cases} \tag{3-10}$$

式中α代表旋转角度，$\cos(\beta_x)$、$\cos(\beta_y)$和$\cos(\beta_z)$表示定位旋转轴的方向余弦。由后 3 个数组成的向量代表的是方向余弦，用来设定特征矢量即固定旋转轴；第 1 个标量数则是绕着该固定旋转轴所旋转过去的角度值。这样的四个数字组成的一组数就称作四元数，而三维空间内的任意旋转都可以简单而精确地用这样的一个四元数组来表示。由于四元数组表示法能减少代码编写者所需做的工作，并且可以减小舍入误差，同时在四元数与四元数之间的插值简单易行等这些种种优势，通过四元数方法来计算旋转已经基本取代了过去的使用方向余弦的方法。

在本程序中，由于需要将某一圆柱体从原点处作坐标变换平移并三维旋转至该圆柱体所对应力链的两个端点：点 1 和点 2 处，先将目标位置的最终圆柱平移到原点处，并将原圆柱沿(a_m, b_m, c_m)对应单位向量旋转 180 度即可，如图 4-9 所示：

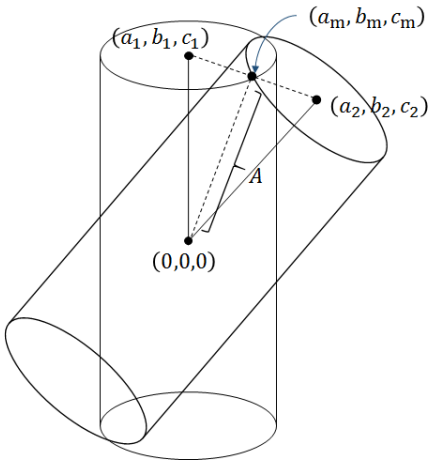

图 3-6 圆柱的三维旋转示意图

其中

$$a_1 = 0, b_1 = \frac{h}{2}, c_1 = 0 \tag{3-11}$$

$$a_2 = \frac{x_2 - x_1}{2}, b_2 = \frac{y_2 - y_1}{2}, c_2 = \frac{z_2 - z_1}{2} \tag{3-12}$$

两者中点坐标

$$a_m = \frac{a_1 + a_2}{2}, b_m = \frac{b_1 + b_2}{2}, c_m = \frac{c_1 + c_2}{2} \tag{3-13}$$

故模量

$$A = \sqrt{{a_m}^2 + {b_m}^2 + {c_m}^2} \tag{3-14}$$

单位向量

$$(x_0, y_0, z_0) = \frac{(a_m, b_m, c_m)}{A} \tag{3-15}$$

故所需求的四元数为$(180°, x_0, y_0, z_0)$。

3.4 速度场云图（Velocity Field）

3.4.1 时刻速度场云图

为显示时刻速度场云图，假设所读取文本文件中的速度位于第 10,11 及 12 列，则在时刻t_j(j表示第j帧)所对应的一个文本文件 file00000j.txt 中可以作一循环，循环次数为文本行数（即颗粒个数）。每次循环计算某行中第 10,11,12 列的平方和并开根，即可获得在该时刻下该行所对应颗粒的速度模量大小，记作$v_{i,j}$(i表示第i个颗粒)。记总颗粒数为n_{par}，则可获得向量

$$[v_{1,j} \quad v_{2,j} \quad v_{3,j} \quad \cdots\cdots \quad v_{n_{par},j}]^T \tag{3-16}$$

为了将该速度向量反映为云图形式，即为所有颗粒根据速度大小赋予色谱梯度，将该向量内部所有数值值中的最大值及最小值作为红色及紫色所对应的速度值，并将其余值依次按照大小排列 RGB 颜色。为实现这一点，首先需要使用算法找出某个 ArrayList（Java 中的动态数组数据结构）中的浮点数最大值 vmax 和最小值 vmin，其函数 findMax 及 findMin 分别写为：

```
public float findMax(ArrayList<Float> arl) {
    float max = arl.get(0);
    int index = 0;
    for (int i = 1; i < arl.size(); i++) {
        if (arl.get(i) > max) {
            max = arl.get(i);
            index = i;
        }
    }
    return arl.get(index);
}
```

以及

```
public float findMin(ArrayList<Float> arl) {
    float min = arl.get(0);
    int index = 0;
    for (int i = 1; i < arl.size(); i++) {
        if (arl.get(i) < min) {
            min = arl.get(i);
            index = i;
        }
    }
    return arl.get(index);
}
```

为了将时刻t_j所有速度$v_{i,j}$对应为相应的 RGB 颜色，采用图 4-10 所示方式进行转换，即规定将 vmax 和 vmin 映射到数字 5 和 0，其余速度值则在 0 和 5 中间依次线性排开，通过这种方式可以将每一个速度值对应到唯一一个 RGB 颜色上。

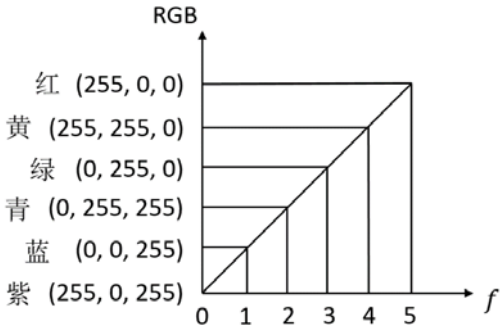

图 3-7 特征值 f 对应 RGB 颜色转换关系函数图

具体代码实现如下所示，首先需要将速度转换为特征值f，由以下代码实现：

```
float vthis = ColumnVelocity.get(i);
float f = 0;
if (vmax != vmin) {
```

```
        f = (0 + (vthis - vmin) * (5 - 0) / (vmax - vmin));
}
if (vmax == vmin) {
        f = 0;
}
```

其中 ColumnVelocity 为速度向量。为防止 vmax = vmin 的情况下分母为零 f 无法计算，实行分类讨论。接着将特征值f转换为 RGB 颜色，由以下代码实现：

```
if (f >= 0 & f <= 1) {
        Color3f c = new Color3f(new Color(255 - (int) (255 * f), 0, 255));
        material1.setDiffuseColor(c);
        appearance.setMaterial(material1);
}
if (f >= 1 & f <= 2) {
        Color3f c = new Color3f(new Color(0, (int) (255 * (f - 1)), 255));
        material1.setDiffuseColor(c);
        appearance.setMaterial(material1);
}
if (f >= 2 & f <= 3) {
        Color3f c = new Color3f(new Color(0, 255, 255 - (int) (255 * (f - 2))));
        material1.setDiffuseColor(c);
        appearance.setMaterial(material1);
}
if (f >= 3 & f <= 4) {
        Color3f c = new Color3f(new Color((int) (255 * (f - 3)), 255, 0));
        material1.setDiffuseColor(c);
        appearance.setMaterial(material1);
}
if (f >= 4 & f <= 5) {
        Color3f c = new Color3f(new Color(255, 255 - (int) (255 * (f - 4)), 0));
        material1.setDiffuseColor(c);
        appearance.setMaterial(material1);
}
```

由于开启速度场云图功能需要修改 Shape3D 几何形体的颜色，而颜色对象的父节点为 Appearance 外观对象，故应在编译前对 Shape3D 对象的行为能力进行设定，将读写其外观的行为设置为 ALLOW，即需在创建 Shape3D 对象时对 Shape3D 对象使用下列函数：

```
setCapability(Shape3D.ALLOW_APPEARANCE_WRITE)
setCapability(Shape3D.ALLOW_APPEARANCE_READ)
```

时刻速度场云图显示效果如图 4-11 所示：

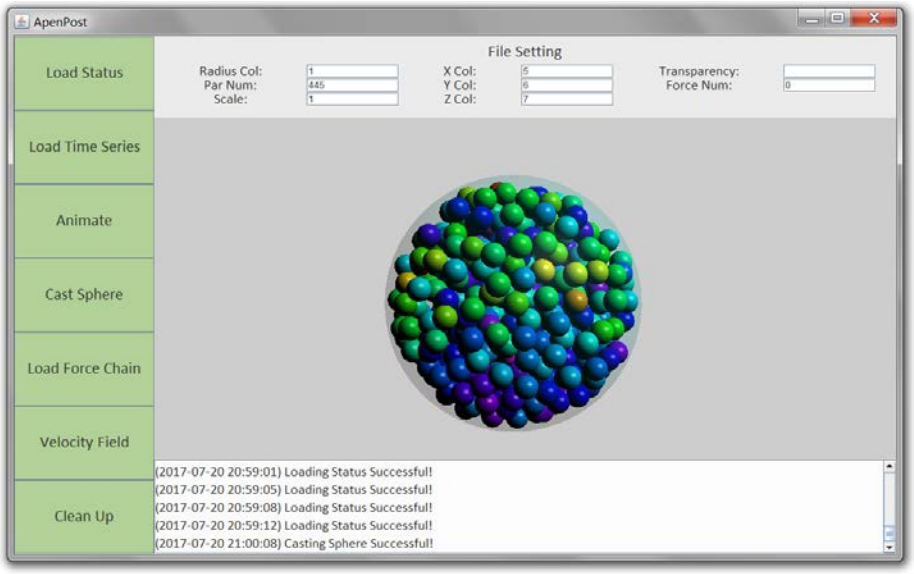

图 3-8 时刻速度场云图显示效果示意图

3.4.2 时间序列速度场云图动画

在时刻速度场云图的基础上，为了将云图以动画形式呈现，设计了基于时间序列颗粒位移动画的时间序列速度场云图动画展示功能。

与时刻速度场云图相同，假设所读取文本文件中的速度位于第 10,11 及 12 列，则在时刻t_j(j表示第j帧)所对应的一个文本文件 file00000j.txt 中可以作一循环，循环次数为文本行数（即颗粒个数）。每次循环计算某行中第 10,11,12 列的平方和并开根，即可获得在该时刻下该行所对应颗粒的标量速度大小，记作$v_{i,j}$(i表示第i个颗粒)。记总颗粒数为n_{par}，则可获得向量

$$[v_{1,j} \quad v_{2,j} \quad v_{3,j} \quad \cdots\cdots \quad v_{n_{par},j}]^T \tag{3-17}$$

记总帧数为n_{frame}，则可以获得由n_{frame}个上述向量所构成的速度张量V，

$$V = [V_1 \quad V_2 \quad V_3 \quad \cdots\cdots \quad V_{n_{frame}}] \tag{3-18}$$

其中

$$\begin{aligned} V_1 &= [v_{1,1} \quad v_{2,1} \quad v_{3,1} \quad \cdots\cdots \quad v_{n_{par},1}]^T \\ V_2 &= [v_{1,2} \quad v_{2,2} \quad v_{3,2} \quad \cdots\cdots \quad v_{n_{par},2}]^T \\ &\cdots\cdots \\ V_i &= [v_{1,i} \quad v_{2,i} \quad v_{3,i} \quad \cdots\cdots \quad v_{n_{par},i}]^T \\ &\cdots\cdots \\ V_{n_{frame}} &= [v_{1,n_{frame}} \quad v_{2,n_{frame}} \quad v_{3,n_{frame}} \quad \cdots\cdots \quad v_{n_{par},n_{frame}}]^T \end{aligned} \tag{3-19}$$

为了实现时间序列速度场云图动画功能，在动画按键 AnimateTimeSeiresButton 收到信号触发动画行为时，初时刻的颗粒除需要进行位移的调整即平移变换之外，还需要改变其 Shape3D 对象的颜色。与时刻速度场云

图不同的是，时间序列速度场云图动画中的色谱不是由一个文本文件决定，而是由时间序列所对应的n_{frame}个文本文件决定。在时刻速度场云图功能代码中，需要找出向量(4-17)中的浮点数最值，故在时间序列速度场云图动画中，则需要找出张量(4-18)中的浮点数最值。

为了实现这一点，将 findMin 和 findMax 方法连续套用两次即可找出整个张量中的浮点数最大值和最小值。实现代码如下，

```
ArrayList<Float> temp = new ArrayList<Float>();
for (int i = 0; i < velocity.size(); i++) {
      float m = findMin(velocity.get(i));
      temp.add(m);
}
vmin = findMin(temp);
```

以及

```
ArrayList<Float> temp2 = new ArrayList<Float>();
for (int i = 0; i < velocity.size(); i++) {
      float m = findMax(velocity.get(i));
      temp2.add(m);
}
      vmax = findMax(temp2);
```

其中 velocity 即速度张量V。

将速度转换为特征值f，以及将特征值f转换为 RGB 颜色的步骤与时刻速度场云图中的实现方法类同，此处不加赘述。需要特别注意的是，由于实现时间序列速度场云图动画功能必须在不同帧数之间不断切换 Shape3D 几何形体的颜色，而颜色对象的父节点为 Appearance 外观对象，应在编译前对 Shape3D 对象的行为能力进行必要的设定，将读写其外观的行为设置为 ALLOW，即需在创建 Shape3D 对象时对 Shape3D 对象使用下列函数：

```
setCapability(Shape3D.ALLOW_APPEARANCE_WRITE)
setCapability(Shape3D.ALLOW_APPEARANCE_READ)
```

切换颗粒外观（颜色父节点）的代码如下：

```
ArrListSphF1.get(i).setAppearance(appearance);
```

其中 appearance 为修改后（即现在帧）的颜色所属的 Appearance 对象，ArrListSphF1 类型为 Sphere 的动态数组，其存放的是有限个数个圆形颗粒。在此情况下 ArrListSphF1 存放的是n_{par}个圆形颗粒，即 ArrListSphF1 的 size 为n_{par}。

需要注意的是，在使用该功能时，用户在通过 Load Time Series 按键加载时间序列文件完毕后，需将 Velocity Field（速度场）按键开启（具体方法为：单击该按键使其背景由灰色切换至绿色）并点击 Animate（动画生成）按键以生成速度场云图动画。

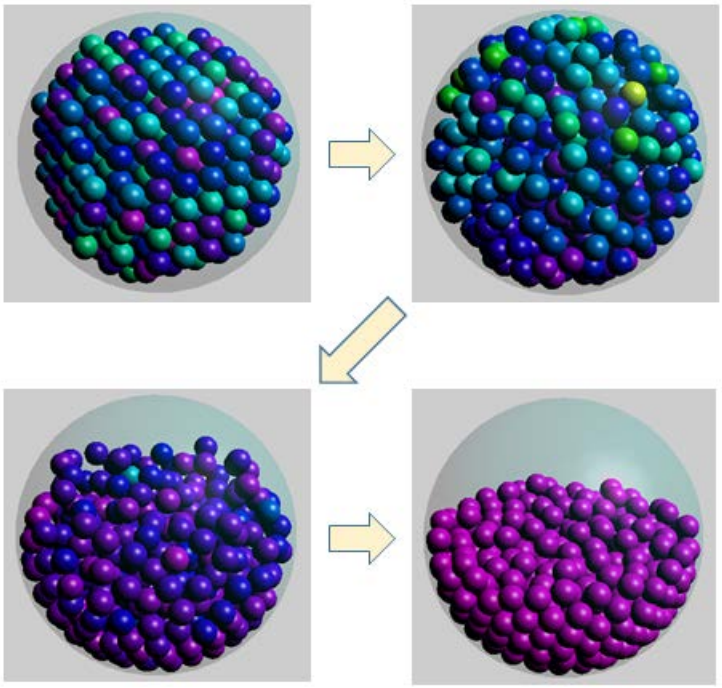

图 3-8 时间序列速度场云图动画显示效果示意图

创作贡献说明

本软件的创作由黄骏教授起意，主要设计和构思由黄骏教授与我（姚凯）共同负责，开发编程及调试由我主要负责。在此感谢黄骏教授及其研究室成员对该软件的设计和开发提出的宝贵意见和建议。