1. 轻量化内核

1.1. 引擎概述

轻量化内核是一款基于 vtk 封装的专业工具，主要用于对有限元模型进行数据处理，支持多种数据过滤抽取方式，生成轻量化的可视化数据，为渲染模块提供高效的数据支持。

该引擎的输入数据为 vtk 对象，支持多种 vtk 数据格式，能够满足不同场景下的有限元模型数据输入需求。其核心目标是通过轻量化处理大幅减少数据量，从而提高数据传输和渲染效率。

轻量化内核输出包含模型的渲染数据和物理场的仿真数据，可以渲染出精美的几何模型和云图效果。

1.2. 核心功能

1.2.1. 多格式数据输入

轻量化内核支持多种数据的输入：vtk、vtu、stl、openFoam 等文件格式，也可以在应用程序构建 vtkUnstructuredGrid 对象，通过 API 输入轻量化内核。

1.2.2. 多样化数据抽取方式

轻量化内核支持多种数据抽取方式：外表面、切片、裁剪、阈值、等值面、等值体。

1.2.3. 数据轻量化处理

轻量化内核会根据数据可视化的要求，采用适当的过滤器，对原始有限元模型数据进行过滤和精简，去除不必要的冗余信息，在保证可视化效果的前提下，大幅减少数据量。

1.2.4. 数据输出

轻量化内核会对抽取结果做离散化处理，输出的数据可以直接用于 3D 渲染。同时保留了仿真场数据，可用于渲染云图效果。

1.3. 技术架构

1.3.1. 底层封装层

基于 vtk 进行封装，利用 vtk 强大的数据处理能力，为引擎提供基础的功能支持。该层主要负责与 vtk 对象的交互，包括数据的读取、解析、处理等操作，将 vtk 的复杂接口进行封装，向上提供简洁、易用的接口。

1.3.2. 数据处理层

1. 数据输入模块：负责接收输入的 vtk 对象数据。
2. 数据抽取模块：实现多种数据抽取算法，提取出所需的轻量化数据。
3. 数据输出模块：对抽取后的数据进行可视化处理，离散为可渲染数据，并关联仿真场数据，组织数据结构并压缩输出。

1.4. 使用流程

1. 数据输入：应用软件调用轻量化 API，传入 vtk 文件或构建好的 vtk 数据对象。
2. 轻量化抽取：根据可视化需求，应用软件设置希望执行的抽取，指定抽取类型及相关参数（比如外表面，或切片位置和方向），调用轻量化 API 执行数据抽取。
3. 数据输出：轻量化内核对抽取后的数据进一步处理，返回可渲染数据及关联的物理场数据。应用软件用这些数据进行 3D 渲染。

1.5. 轻量化内核 API

1.5.1. open

打开一个求解结果文件，准备用于数据抽取。
public LWResult open(String filename, String filetype)

请求参数
- filename：文件名
- filetype：文件类型（可选）

返回数据
- LWResult

示例：
```
{
  'ver': '1.0',
  'errno': 0,
  'out': {
     'id': id,              // uuid，唯一对应打开的数据
     'properties': properties, // 概要信息数据
     'steps': steps         // 帧数据（如果只有一帧，值为 null）
  }
}
```

1.5.2. openWithData

打开一个求解结果数据，准备用于数据抽取。
public LWResult openWithData(List<vtkDataObject> dataSets, List<Double> steps, List<FrameCustData> custDatas, Integer fsMode)

请求参数
- dataSets：一组 VTK 数据对象，对应一系列帧的数据，通常是 List<vtkUnstructuredGrid>
- steps：一系列帧的时间
- custDatas：一组自定义数据，用于传递额外的数据，对应一系列帧的数据
- fsMode：指定过滤几何数据，还是仅提取场数据，以优化性能

返回数据
- LWResult

示例：
```
{
  'ver': '1.0',
  'errno': 0,
  'out': {
    'id': id,              // uuid，唯一对应打开的数据
    'properties': properties, // 概要信息数据
    'steps': steps         // 帧数据（如果只有一帧，值为 null）
  }
}
```

1.5.3. close

关闭一个求解结果数据，释放内存。
public LWResult close(String id)

请求参数
- Id：之前打开的数据 id

返回数据
- LWResult

示例：
```
{
  'ver': '1.0',
  'errno': 0,
  'out': {
  }
}
```

1.5.4. filter

对已经打开的数据进行轻量化抽取。
public LWData filter(String id, int index, List<Map<String, Object>> parametersSet, List<String> scalarNames, Integer syncMode, Integer filterMode, Integer fsMode, List<FSInfo> fsInfos)

请求参数
- Id：之前打开的数据 id
- Index：帧数据 List 的索引
- parametersSet：抽取参数，指定一个或多个抽取类型及其参数。具体参见下面的示例。
- scalarNames：指定要抽取和返回的场数据名称
- syncMode：1：同步执行；2：多部件并行执行
- filterMode：1：输入的场数据关联在节点上；2：输入的场数据是：单元+节点 -> 一个场数据

- fsMode：1：处理几何和场数据；2：处理几何数据：3：处理场数据
- fsInfos：已抽取的外表面的单元和节点信息，模型拓扑不变的情况下加速场数据处理

抽取参数示例：

```
parametersSet = []  // 抽取参数

surfaceParam = {
   'type': 'Surface',
}
parametersSet.append(surfaceParam)

sliceParam = {
   'type': 'Slice',
   'origin': [0, 0, 0],
   'normal': [1, 0, 0]
}
parametersSet.append(sliceParam)

clipParam = {
   'type': 'Clip',
   'origin': [0, 0, 0.002],
   'normal': [0, 0.1, -1]
}
parametersSet.append(clipParam)

crinkleClipParam = {
   'type': 'CrinkleClip',
   'origin': [0, 0.04, 0],
   'normal': [-1, -0.5, -1]
}
parametersSet.append(crinkleClipParam)

contourParam = {
   'type': 'Contour',
   'name': 'pressure',
   'value': 0.4
}
parametersSet.append(contourParam)

isovolumeParam = {
   'type': 'IsoVolume',
   'name': 'pressure',
   'min': 80.7,
   'max': 100.9
}
parametersSet.append(isovolumeParam)

thresholdParam = {
   'type': 'Threshold',
   'name': 'pressure',
   'min': 0.1,
   'max': 0.2
}
parametersSet.append(thresholdParam)
```

返回数据

- 抽取结果具有目录结构，压缩成一个 zip 包。

```
抽取 id1                    // 抽取 id（比如是一个切面）。序号对应抽取请求在数组的 index
 部件 id1                   // 部件 id。序号对应之前调用 Open 打开数据时的 Part 数组的 index
   geo_edge               // wireframe 的索引，二进制 int 序列
   geo_line               // outline 的索引，二进制 int 序列
   geo_normals            // 法线
   geo_poly               // 三角形面片的索引，二进制 int 序列
   geo_positions          // 顶点
   geo_positionId         // 顶点关联的全量网格的顶点 ID，二进制 int 序列
   geo_beam               // 梁单元
   res_pressure           // 场数据，都以"res_"开头
   res_turb_kinetic_energy
   res_velocity_magnitude
   res_viscosity_turb
   bodyid                 // 部件的 bodyid，数据类型 int
抽取 id2                    // 抽取 id（比如是一个切面）。序号对应抽取请求在数组的 index
   ......
```

1.5.5. 数据结构定义

1.5.5.1. LWResult

```
public class LWResult implements Serializable {
    public String ver;
    public Integer errno;
    public LWOut out;
}
```

1.5.5.2. LWOut

```
public class LWOut implements Serializable {
    public String id;
    public List<List<Map<String, Object>>> properties;
    public List<Double> steps;
}
```

1.5.5.3. FSInfo

```
public class FSInfo implements Serializable {
    public long[] nodeIds;
    public CellNode[] cellNodes;
}
```

1.5.5.4. CellNode

```
public class CellNode implements Serializable {
```

```
    public long cellId;
    public long nodeId;
}
```