



江南大学  
JIANGNAN UNIVERSITY

物联网工程学院

面向对象程序设计/Java 课程期末作业报告

网页端三效蒸发反应模拟程序设计

班 级：物联 1601

姓 名：尹达恒

学 号：1030616134

指导老师：陈树

2018~2019 第一学期

2019 年 3 月 6 日

## 摘要

三效蒸发反应广泛应用于食品、化工、医药、生物和环保领域，如牛奶、果汁、盐、糖、抗生素、氨基酸、废水的处理。由于该过程发生汽化和冷凝等相变化，蒸发过程需要消耗大量的蒸汽用于加热，装置设计、操作的优劣在很大程度上影响反应的效果。因此，如果能用计算机对三效蒸发反应进行模拟和测试，将能有效地提升三效蒸发反应器的设计和测试和维护成本，为化工厂节约大量资源。本文所介绍的三效蒸发反应模拟程序使用网页开发中常用的 JavaServlet+JavaScript 模式处理模拟数据，结合最近推出的高性能网端 3D 图像引擎 three.js 进行呈现，使得对三效蒸发反应的模拟过程更加快速、结果更加直观，为三效蒸发反应器的设计和测试带来极大便利。

## 目录

1 设计要求 .....	3
1.1 后端设计要求 .....	3
1.2 前端设计要求 .....	3
2 设计内容 .....	4
2.1 后端设计内容 .....	4
2.2 前端设计内容 .....	4
2.2.1 前端 JavaScript 第三方库文件 .....	4
2.2.2 前端 JavaScript 程序文件及函数功能 .....	5
2.2.3 三效图初始化过程 .....	7
2.2.4 三效图动画子进程运行原理 .....	7
3 程序运行截图 .....	8

## 1 设计要求

### 1.1 后端设计要求

后端用 Java Servlet 编写，要求能接收前端发送的数据以及计算模拟结果。

### 1.2 前端设计要求

前端用 JavaScript 编写，包含以下内容：

- 数据交互：能与后端进行数据交互。
- 三效图主体：在页面上以图像方式表现三效图模拟结果，并能根据模拟数据实时更新显示。
- 数据表格：在表格中以数值方式列出三效图模拟结果，并能根据模拟数据实时更新显示。
- 三效图参数调节滑动条：能对三效模拟的液体和气体输入进行调节。
- 三个按钮，功能如下：
  - 开始：
    1. 三效开始进料，到 50% 时三效输出阀门打开，物料进入一效，同时一效蒸汽阀打开；
    2. 一效液位到达 50% 时，一效输出阀门打开，物料进入二效，同时二效蒸汽阀打开；
    3. 二效液位达到 50% 时，二效输出阀门打开，开始成品输出以及半成品回收。
  - 保存：将当前三效模拟数据以文本文件方式保存在本地。
  - 读取：从保存的三效模拟数据文件中读取三效模拟数据并更新模拟过程。

## 2 设计内容

### 2.1 后端设计内容

后端数据交互步骤如下：

1. post 方式接收前端传来的数据，并从中读取液体输入速率 liq\_in、气体输入速率 gas\_in、液位限制 hMax、压强限制 pMax、液体输入速率限制 vlMax、气体输入速率限制 vpMax；

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws IOException
{
    double[] p = new double[]{0, 0, 0};
    double[] h = new double[]{0, 0, 0};
    double[] vl = new double[]{0, 0, 0};
    double[] vout = new double[]{0, 0};
    double[] vp = new double[]{0, 0, 0};

    double hMax = Double.valueOf(request.getParameter("hMax"));
    double vlMax = Double.valueOf(request.getParameter("vlMax"));
    double vpMax = Double.valueOf(request.getParameter("vpMax"));
    double pMax = Double.valueOf(request.getParameter("pMax"));
    vl[0] = Math.min(Double.valueOf(request.getParameter("liq_in")), vlMax);
    vp[0] = Math.min(Double.valueOf(request.getParameter("gas_in")), vpMax);
```

图 1. 后端程序截图：接收数据

2. 根据接收到的数据计算三效图各环节的模拟结果；

```
for (int i = 0; i < 2; i++)//算v
{
    vl[i + 1] = vLNext(vl[i], vlMax);
    vp[i + 1] = vpNext(vp[i], vpMax);
}
h[2] = vLToH(vl[0], vlMax, hMax);
h[0] = vLToH(vl[1], vlMax, hMax);
h[1] = vLToH(vl[2], vlMax, hMax);
p[0] = vpToP(vp[0], vpMax, pMax);
p[1] = vpToP(vp[1], vpMax, pMax);
p[2] = vpToP(vp[2], vpMax, pMax);

vout[0] = Math.min(p[1] / pMax, h[1] / hMax) * vlMax;
vout[1] = Math.abs(p[1] / pMax - h[1] / hMax) * vlMax;
```

图 2. 后端程序截图：处理数据

3. 以 JSON 格式返回计算结果。

```
JSONObject jo = new JSONObject();
jo.element("p", toJSONArray(p));
jo.element("h", toJSONArray(h));
jo.element("vl", toJSONArray(vl));
jo.element("vout", toJSONArray(vout));
jo.element("vp", toJSONArray(vp));

response.getWriter().print(jo.toString());
```

图 3. 后端程序截图：返回结果

### 2.2 前端设计内容

#### 2.2.1 前端 JavaScript 第三方库文件

本项目使用的 JavaScript 库包括：

- jQuery 插件
- 基于 WebGL 图像引擎的 3D 渲染库 three.js
- js 文件保存插件 FileSaver.js

引入的库文件的方式遵循 jQuery 插件和 three.js 库的官方文档。

```
<script src="js/jquery-3.3.1.js"></script>
<script src="js/three.js"></script>
<script src="js/controls/OrbitControls.js"></script>
<script src="js/objects/Reflector.js"></script>
<script src="js/objects/Refractor.js"></script>
<script src="js/objects/Water2.js"></script>
<script src="js/WebGL.js"></script>
<script src="js/renderers/CSS2DRenderer.js"></script>
<script src="js/libs/dat.gui.min.js"></script>
<script src="js/libs/FileSaver.min.js"></script>
```

图 4. 前端程序截图：前端所用的 JavaScript 第三方库文件

### 2.2.2 前端 JavaScript 程序文件及函数功能

前端所用的 JavaScript 程序文件如下图：

```
<script src="3Dvalues.js"></script>
<script src="3Dinit_func.js"></script>
<script src="3Ddata_init_func.js"></script>
<script src="3Dshape_func.js"></script>
<script src="3Danni_func.js"></script>
<script src="3Dmain_anmi.js"></script>
<script src="3Dinit.js"></script>
<script src="3Ddata_handle.js"></script>
```

图 5. 前端程序截图：前端所用的 JavaScript 程序文件

程序文件功能及其中各函数功能如下：

- 3Dvalues.js：记录了绘制 3D 图形所需的必要信息。
- 3Dinit\_func.js：定义了初始化三效图时要调用的函数和初始数据，包括：
  - sceneInit()：初始化 3D 场景并添加地板平面和 3D 背景。
  - lightInit()：在场景中设置灯光。
  - shapeInit()：向场景中添加三效图的 3D 图形并贴图。
  - rendererInit()：设置并开启渲染器。
  - cameraInit()：添加并调整相机视角。
- 3Ddata\_init\_func.js：定义了和后台进行数据交互时要调用的函数和初始数据，包括：
  - guiInit()：初始化液体和气体输入参数调节面板。
  - requestData()：向后端发送液体和气体输入数据，将返回的计算结果传入 updateData(response) 进行处理。
  - updateData(response)：依照传入计算结果 response 中的数据对前端暂存的三效模拟数据更新。

- `start_click()`: 1.2 中所述的“开始”按钮的点击事件所绑定的函数，更新页面显示并调用 `start_anmiate()` 播放 1.2 中所描述的开始动画。
- `3Dshape_func.js`: 定义了用于创建 3D 形状的函数，这些函数会于初始化三效图时被 `3Dinit_func.js` 中定义的 `shapeInit()` 函数调用，包括：
  - `createIOarrow(name)`: 返回一个带有字符串标签的白色圆锥体，标签中的字符串为传入参数 `name`；圆锥体的贴图不受光照影响。返回的圆锥体会被调整到液体或气体输入管道入口或产品输出口处。
  - `createBucket(name)`: 返回一个带有字符串标签的蓝色框架型圆柱体，标签中的字符串为传入参数 `name`。返回的圆柱体即是三效图中的反应釜。
  - `createBucketWater()`: 返回一个由蓝色半透明圆柱体和位于圆柱体上表面的水面反射效果平面组成的 3D 图形组。返回的 3D 图形组即是三效图中反应釜里的液体，图形组会被调整到与某个反应釜在中轴线重合。
  - `calculateCurve(t, length, points)`: 返回一个三元组，其值为点集 `points`(四维数组) 中各点顺序相连所构成的曲线上某点的坐标。传入参数中 `length` 指定了 `points` 构成的曲线长度，`t(0≤t≤1)` 指定了要计算的点在曲线上的位置。此函数只在 `calculateTubeShape(points)` 中被调用。
  - `calculateTubeShape(points)`: 返回无材质曲线管道，曲线管道的中轴线为将点集 `points`(四维数组) 中各点顺序相连所构成的曲线。此函数只在 `createTube(points)` 中被调用。
  - `createTube(points)`: 返回蓝色框架型曲线管道，此管道为在 `calculateTubeShape(points)` 返回的无材质曲线管道上生成蓝色框架贴图而成。返回的管道框架组即是三效图中的液体管道或气体管道。
  - `createTubeWater(points, v_water)`: 返回曲线管道形流体，此流体为在 `calculateTubeShape(points)` 返回的无材质曲线管道上生成水面反射效果贴图而成。返回的流体即是三效图液体管道中流动的液体。`v_water` 指定了水面反射效果中的水体流动速度。
- `3Danmi_func.js`: 定义了用于在场景动画中按照模拟得到的数据更新三效图和数据表格的函数，包括：
  - `updateScene()`: 调用 `updateHP(h,p)`、`updateCubeWater(vl,vout)`、`updateInfo(vp, vl, vout)` 根据前端暂存的数据更新界面显示。此函数主要在动画子进程函数 `animate()` 中调用。
  - `updateHP(h,p)`: 更新三效图反应釜中的液位高度和压强。
  - `updateCubeWater(vl, vout)`: 更新三效图管道中的气体或液体流动速度。
  - `updateInfo(vp, vl, vout)`: 更新页面上数据表格里的数据。
- `3Dmain_anmi.js`: 定义了场景动画的两个主要函数，包括：
  - `animate()`: 场景动画子进程，循环调用自身。在函数体中调用 `updateScene()` 更新界面显示。
  - `start_anmiate()`: 播放 1.2 中所描述的开始动画。只在 `start_click()` 函数中被调用。
- `3Dinit.js`: 调用上述文件中的函数执行页面初始化操作。
- `3Ddata_handle.js`: 定义了用于保存数据文件和读取数据文件的函数，包括：
  - `handleFiles(files)`: 传入参数 `files` 为上传的数据文件，读取文件中的信息并更新显示。
  - `saveData()`: 将当前数据保存为 JSON 格式的文本文件。

### 2.2.3 三效图初始化过程

1. 3Dvalues.js 中定义并初始化所有 3D 图形的大小和位置数组。
2. 3Dmain\_anmi.js 中调用 sceneInit() 初始化 3D 场景。
3. 3Dmain\_anmi.js 中调用 lightInit() 初始化灯光。
4. 3Dmain\_anmi.js 中调用 shapeInit() 初始化三效图的形状和贴图。
5. 3Dmain\_anmi.js 中调用 rendererInit() 设置并开启场景渲染。
6. 3Dmain\_anmi.js 中调用 cameraInit() 初始化视角位置和方向。
7. 3Dmain\_anmi.js 中调用 animate() 开启动画子进程。

```
sceneInit();
lightInit();
shapeInit();
rendererInit();
cameraInit();
animate();|
```

图 6. 前端程序截图：3Dinit.js 三效图初始化过程

### 2.2.4 三效图动画子进程运行原理

三效图的动画子进程 `animate()` 是独立于主进程运行的，其功能是通过不断读取前端暂存的三效模拟数据和当前显示的三效模拟效果进行比较，按照比较结果对三效图中各 3D 图形的参数进行更新，同时更新数据表格中显示的数值，以达到实时更新的效果。因此每当前端暂存的三效模拟数据发生变化时，动画子进程 `animate()` 中都能检测到变化并及时做出响应调整界面显示。

此外，动画子进程 `animate()` 中更新显示的函数并不是完全按照前端暂存的三效模拟数据立即更新 3D 图形的参数，而是按照当前显示的模拟值和暂存的三效模拟数据的差值进行适当调整，从而一定程度上模拟出实际三效蒸发反应中负反馈调节的效果。

### 3 程序运行截图

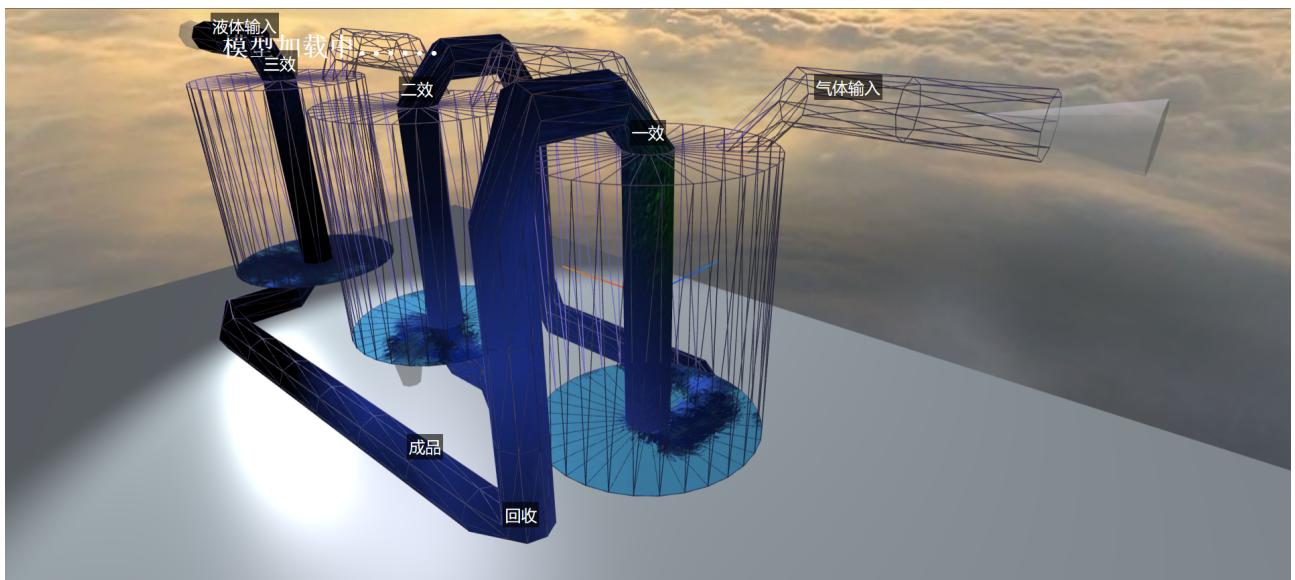


图 7. 程序运行截图：三效图初始化

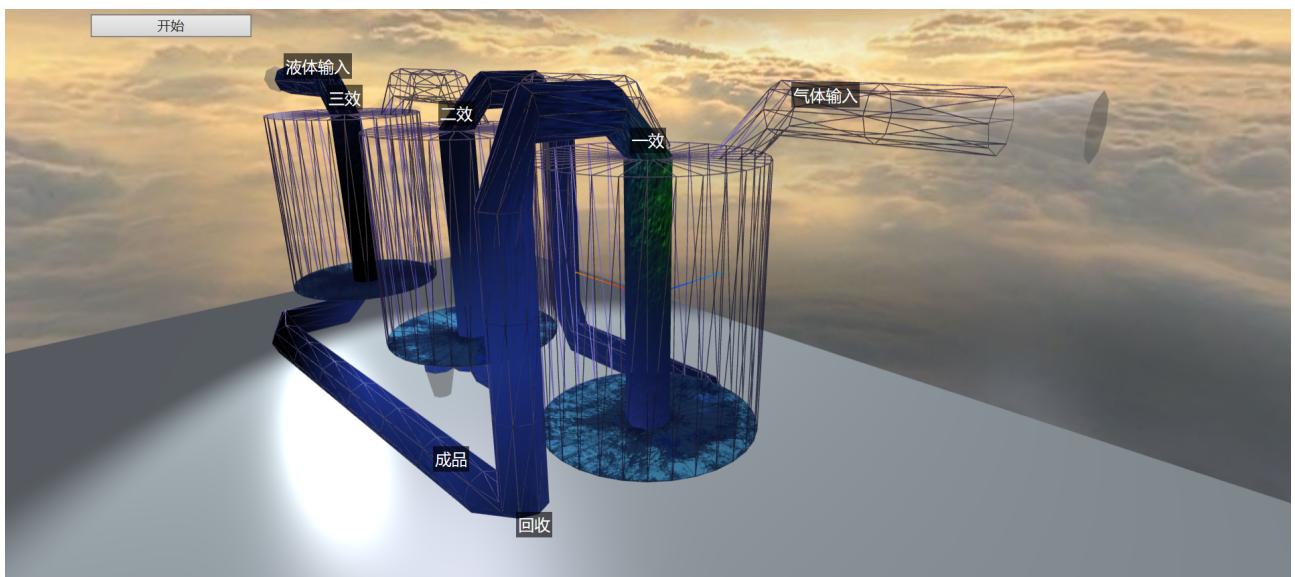


图 8. 程序运行截图：三效图初始化完成

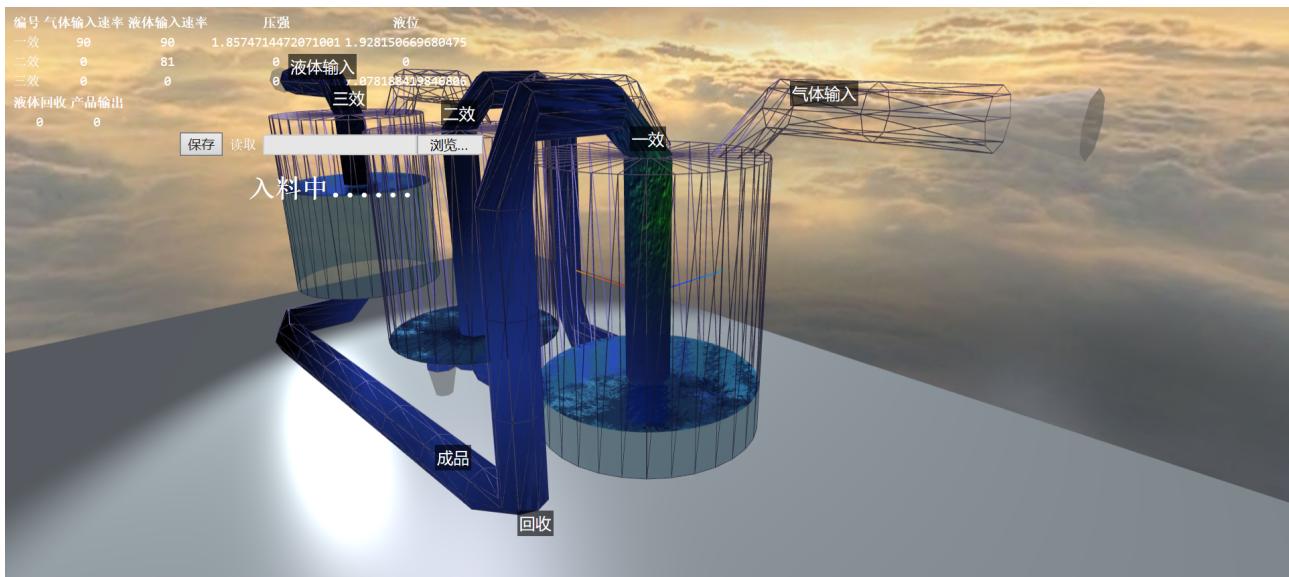


图 9. 程序运行截图：开始入料

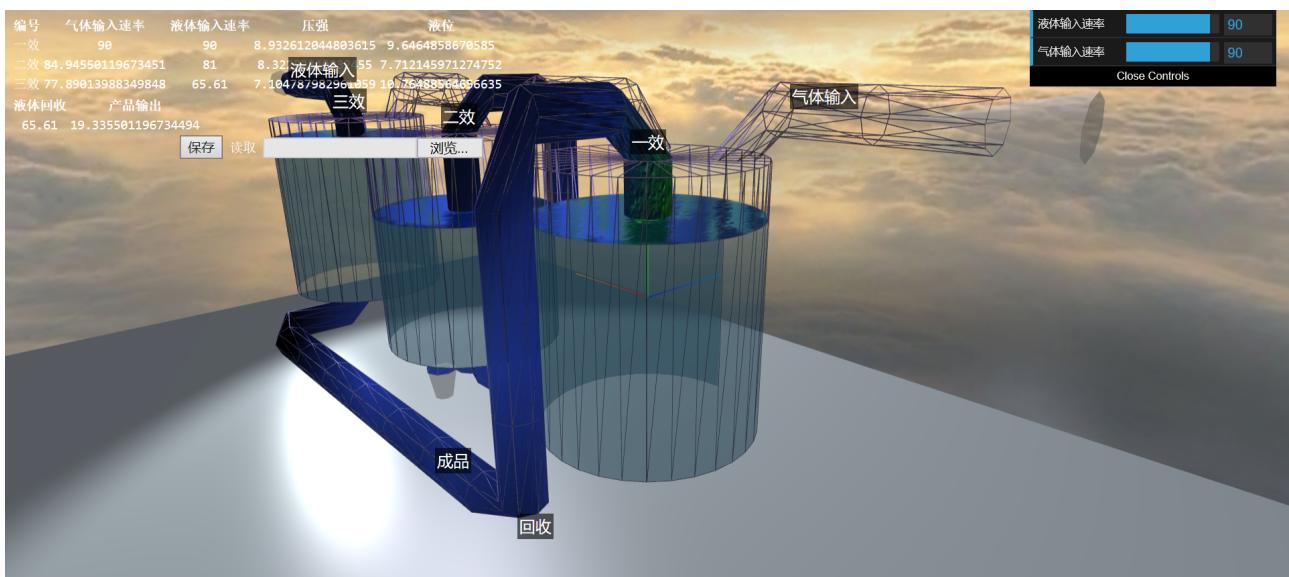


图 10. 程序运行截图：入料完成

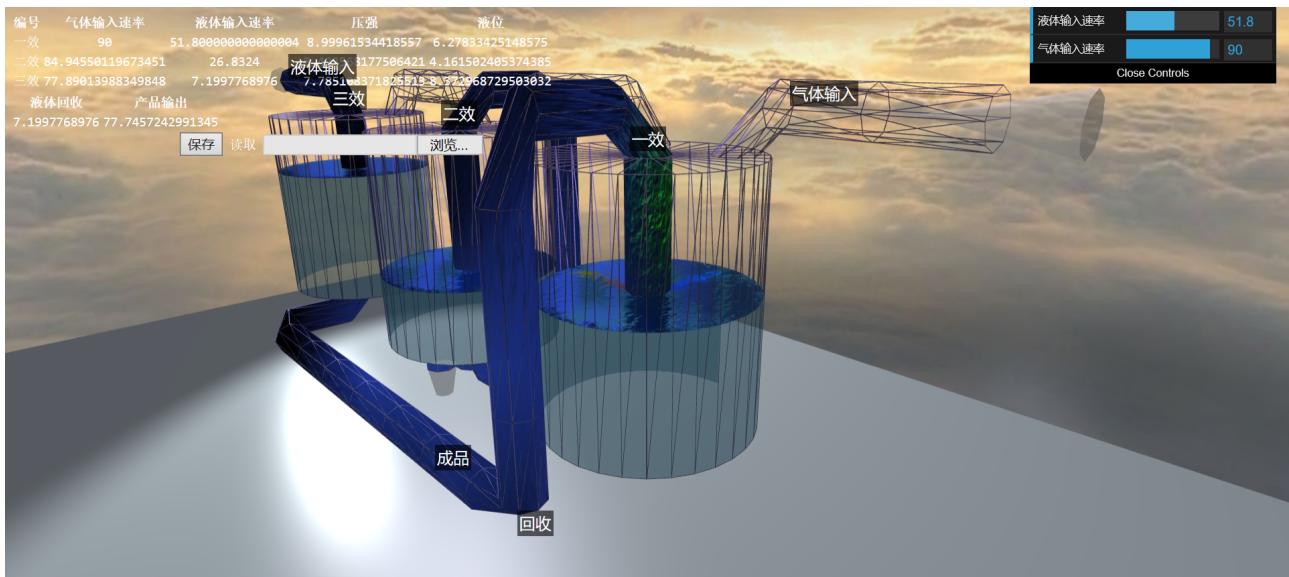


图 11. 程序运行截图：调整液体输入速率

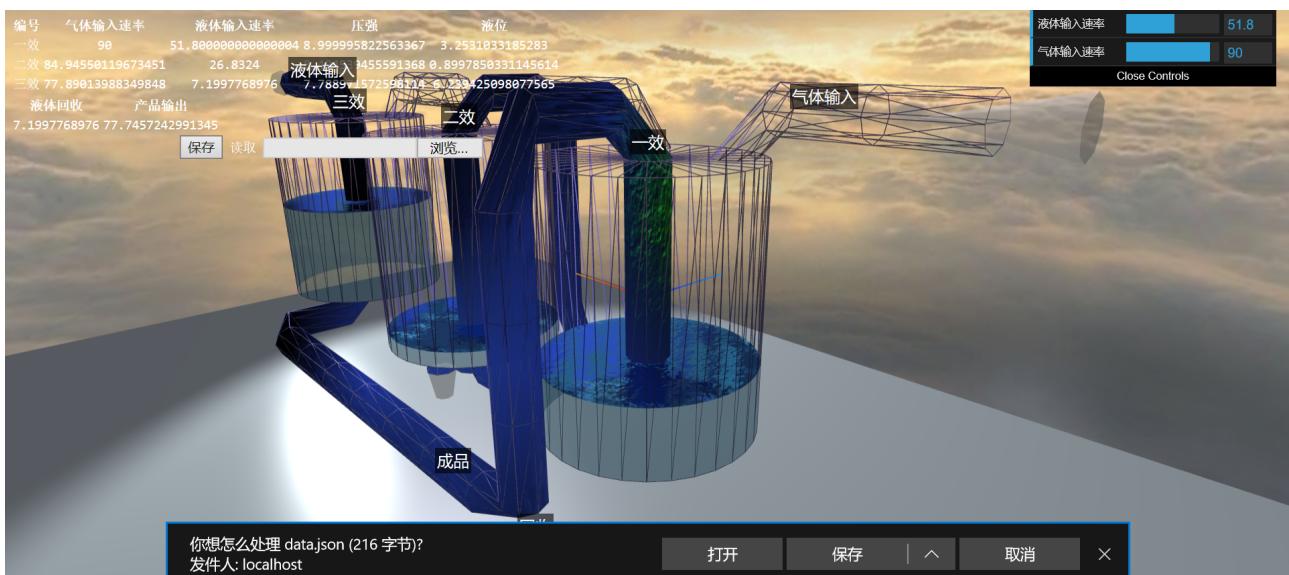


图 12. 程序运行截图：保存数据文件

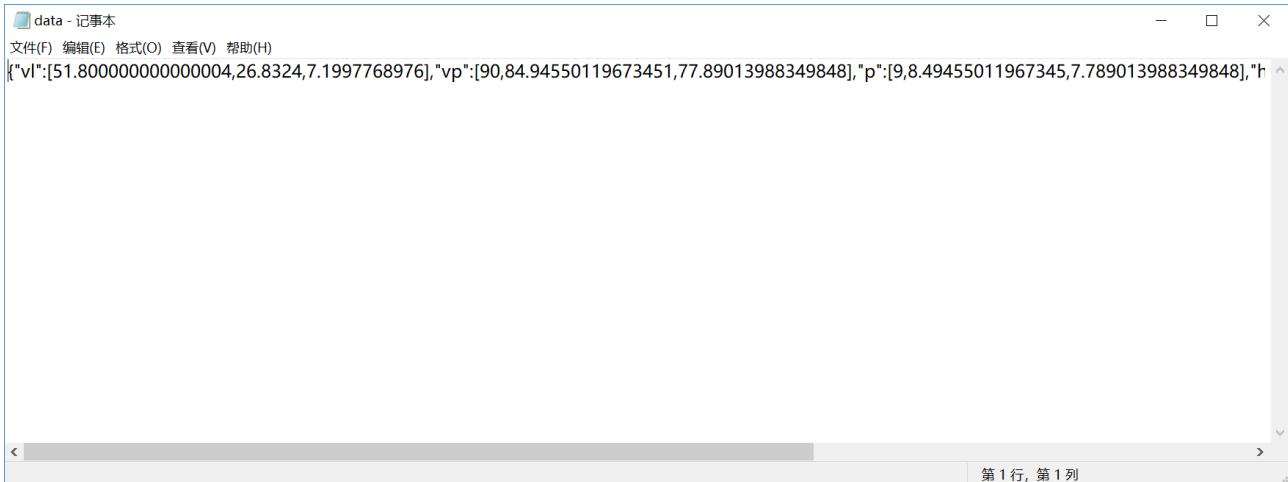


图 13. 程序运行截图：数据文件格式

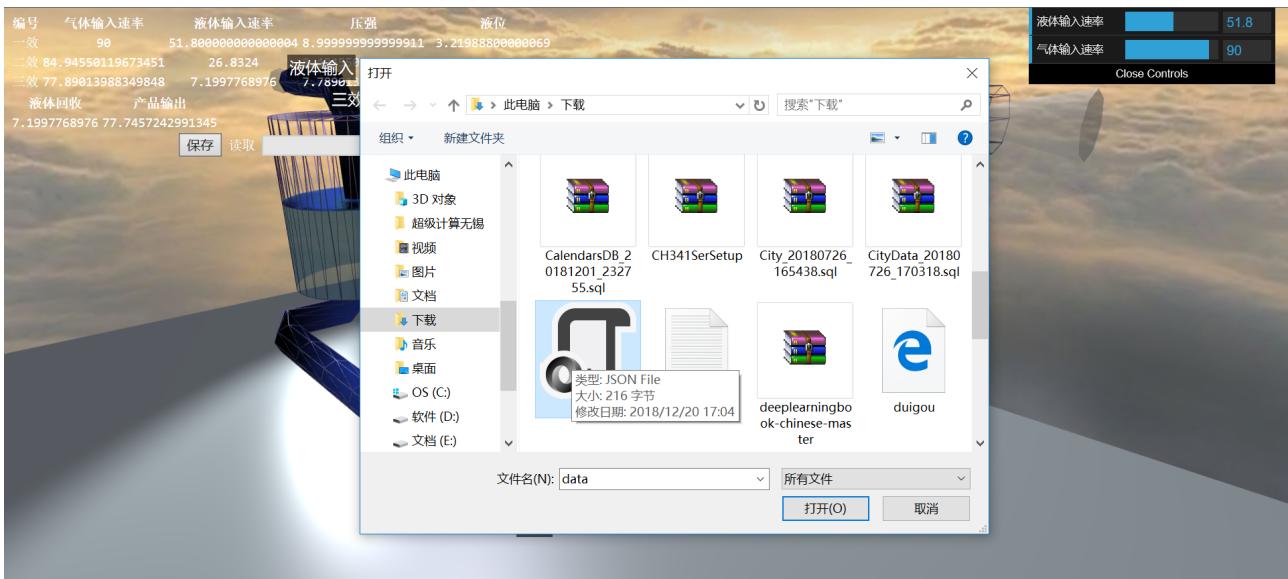


图 14. 程序运行截图：读取数据文件