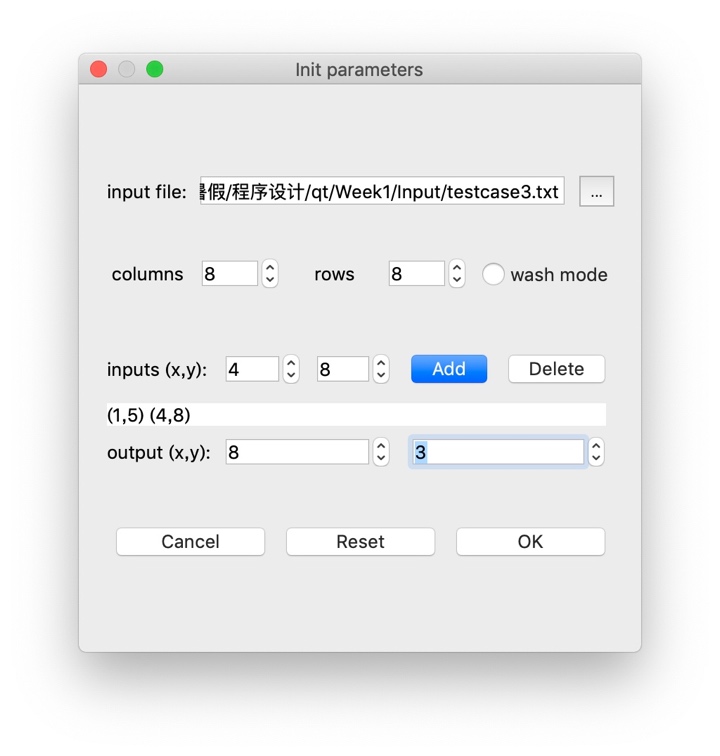
**DMFB 模拟系统：实验报告**

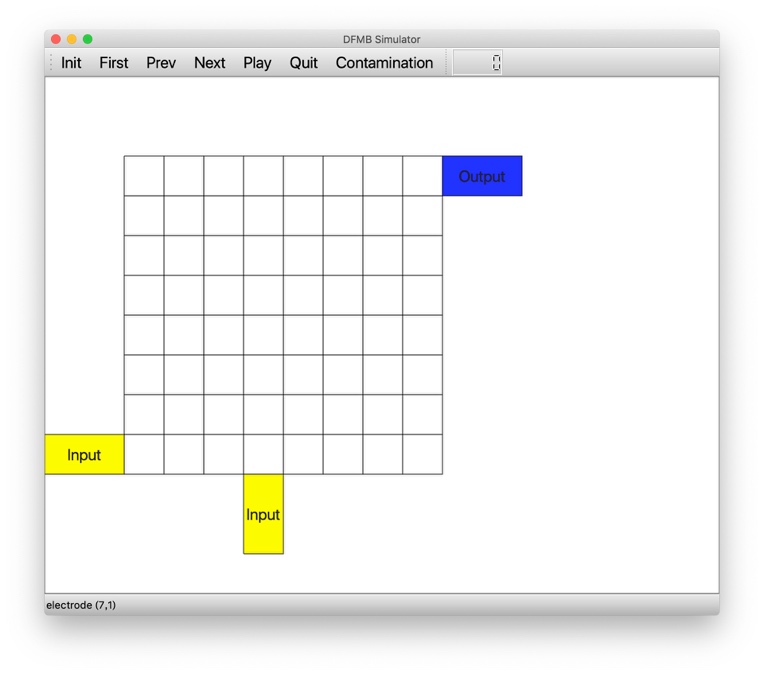
2018011359 计84 乐阳

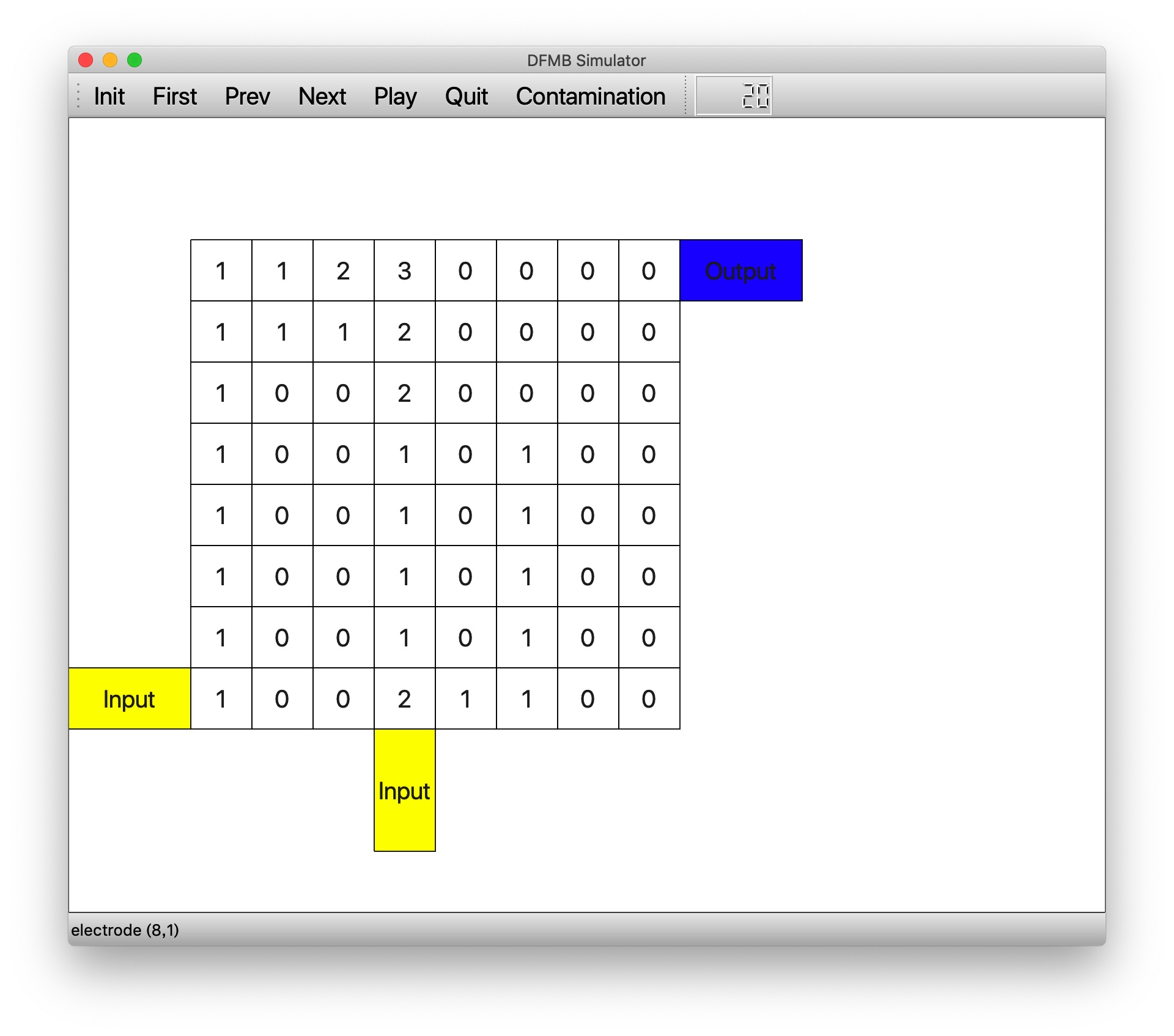
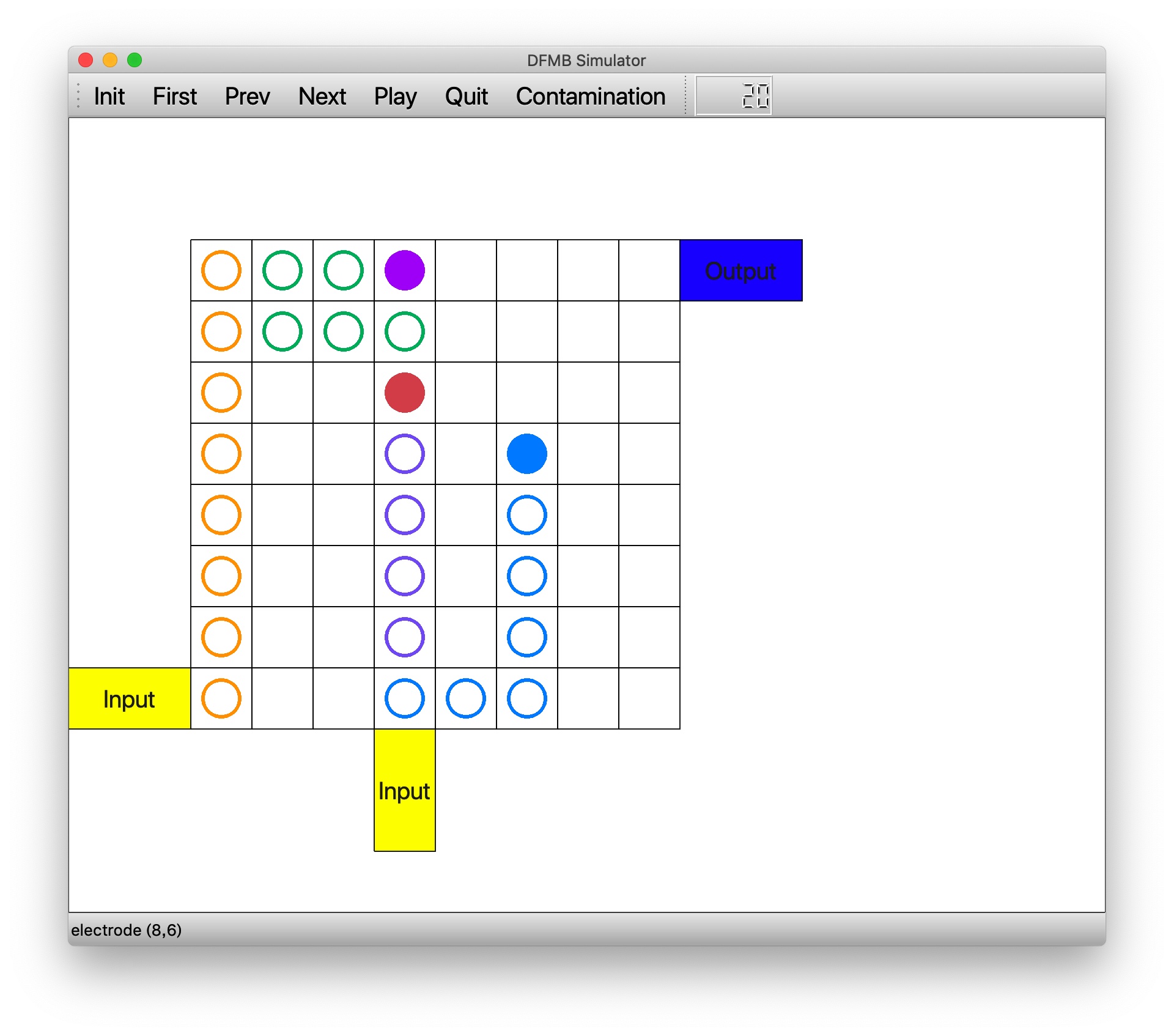
1. **功能展示**
2. 初始化对话框

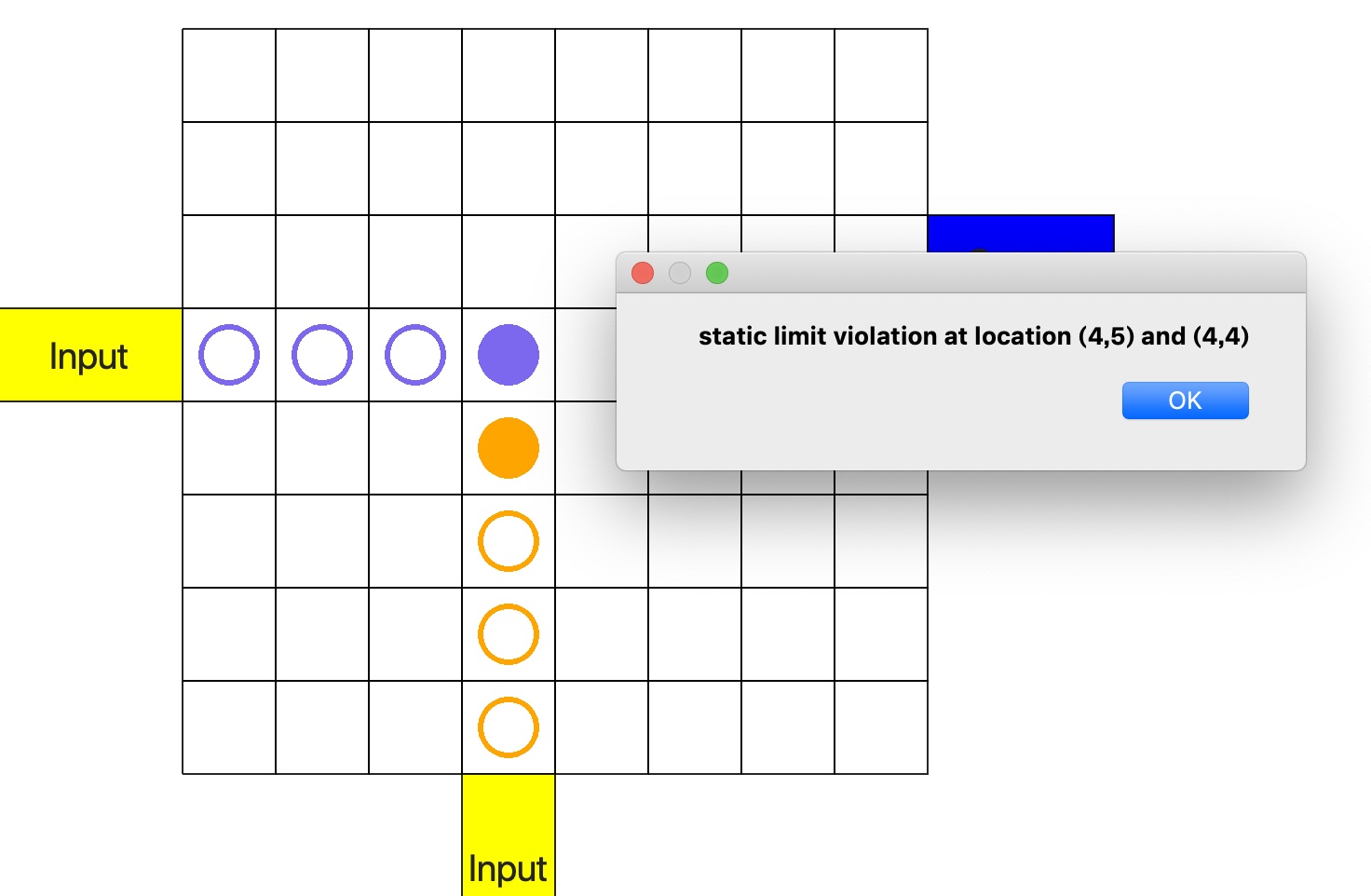
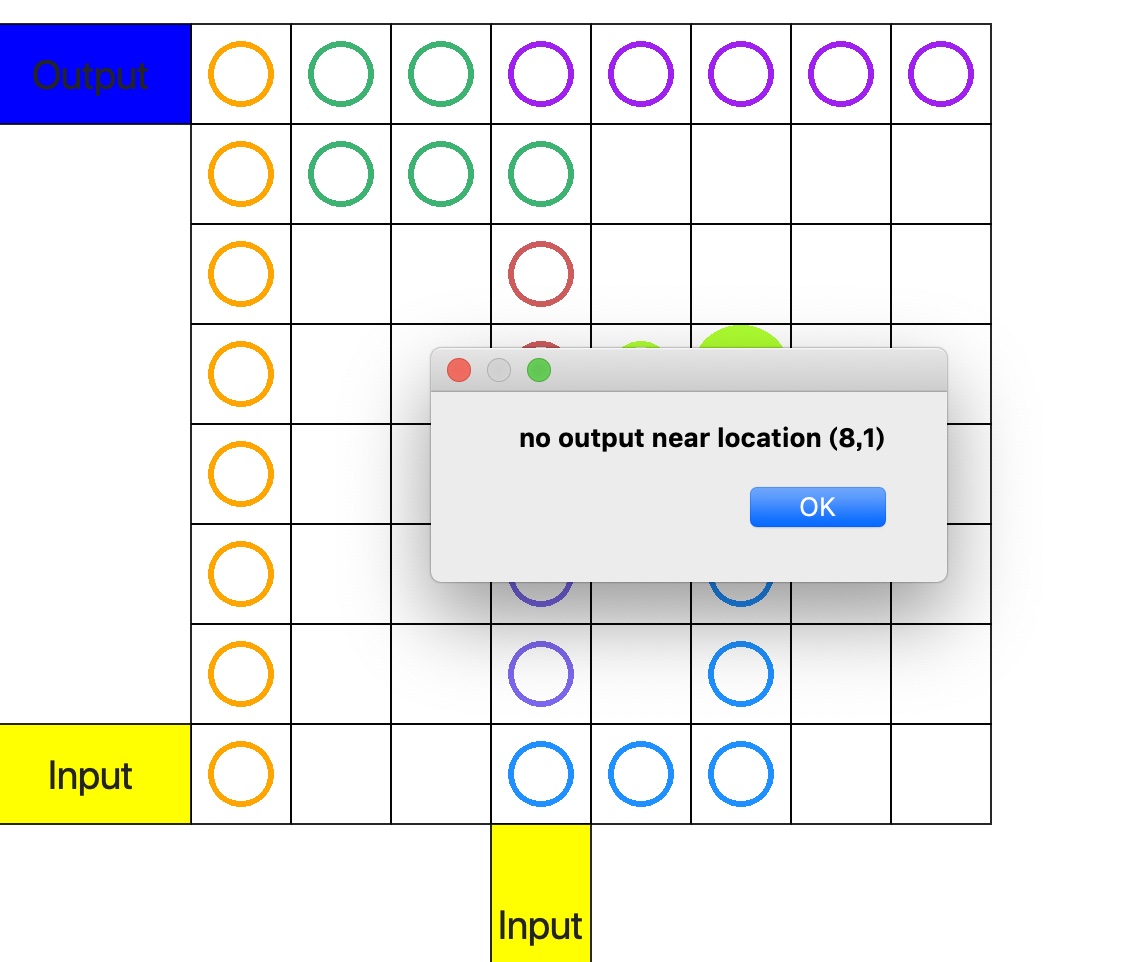


* 可以点击一个ToolButton弹出文件对话框选择读入的文件
* 可点击wash mode按钮设置清洗模式
* 键入列数、行数。同时小于等于3时警告
* 键入输入端口的坐标，点击右侧的Add和Delete按钮可添加/删除端口，输入的数据显示在下面的QLabel上
* 输入、输出端口检查报错：端口必须位于网格边界，如果开启清洗模式则端口不能位于左下和右上（与清洗液滴输入和回收位置冲突）
* 按Reset键可重新输入以上内容，OK键载入设置进入主界面。

1. 主界面

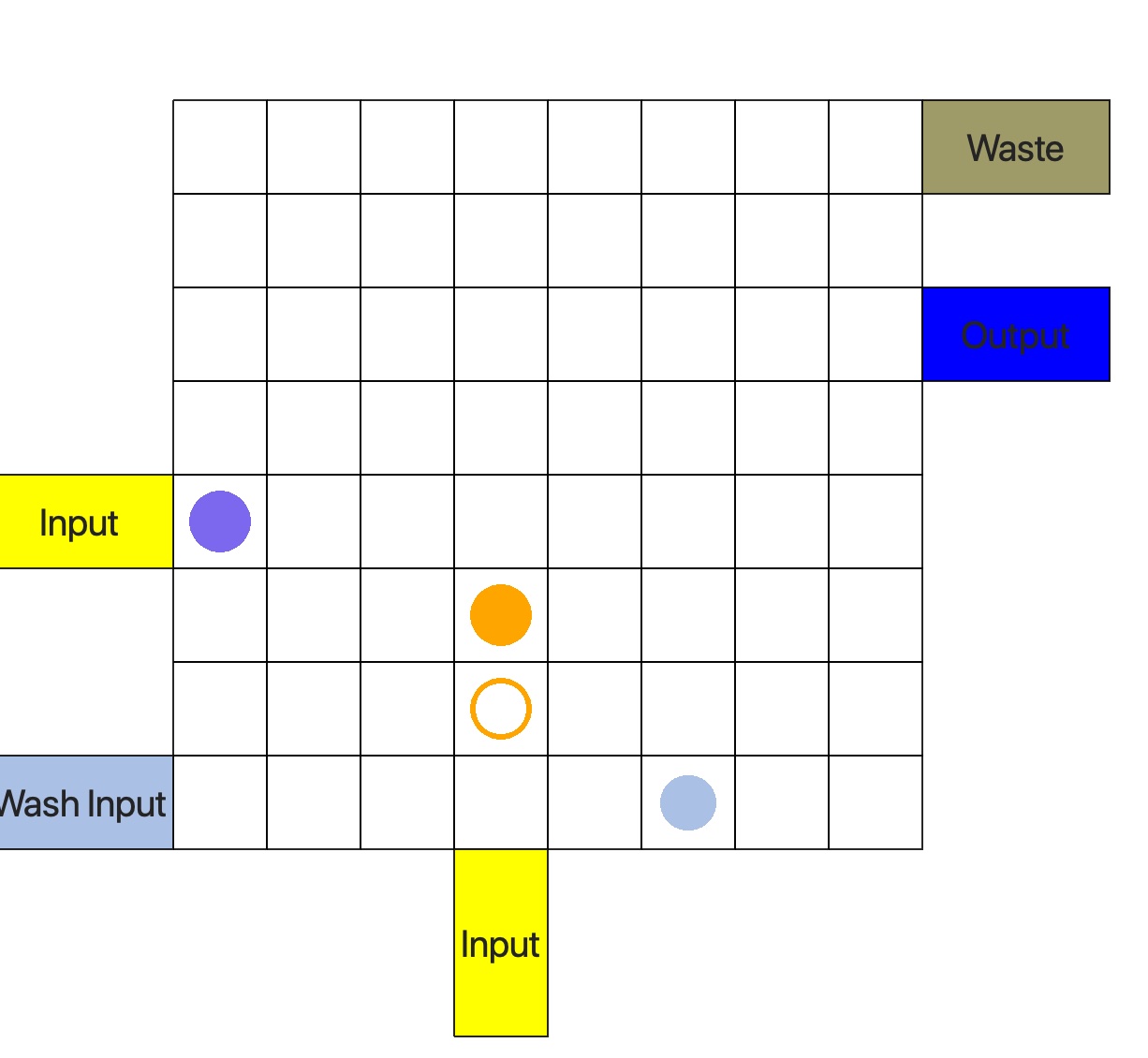
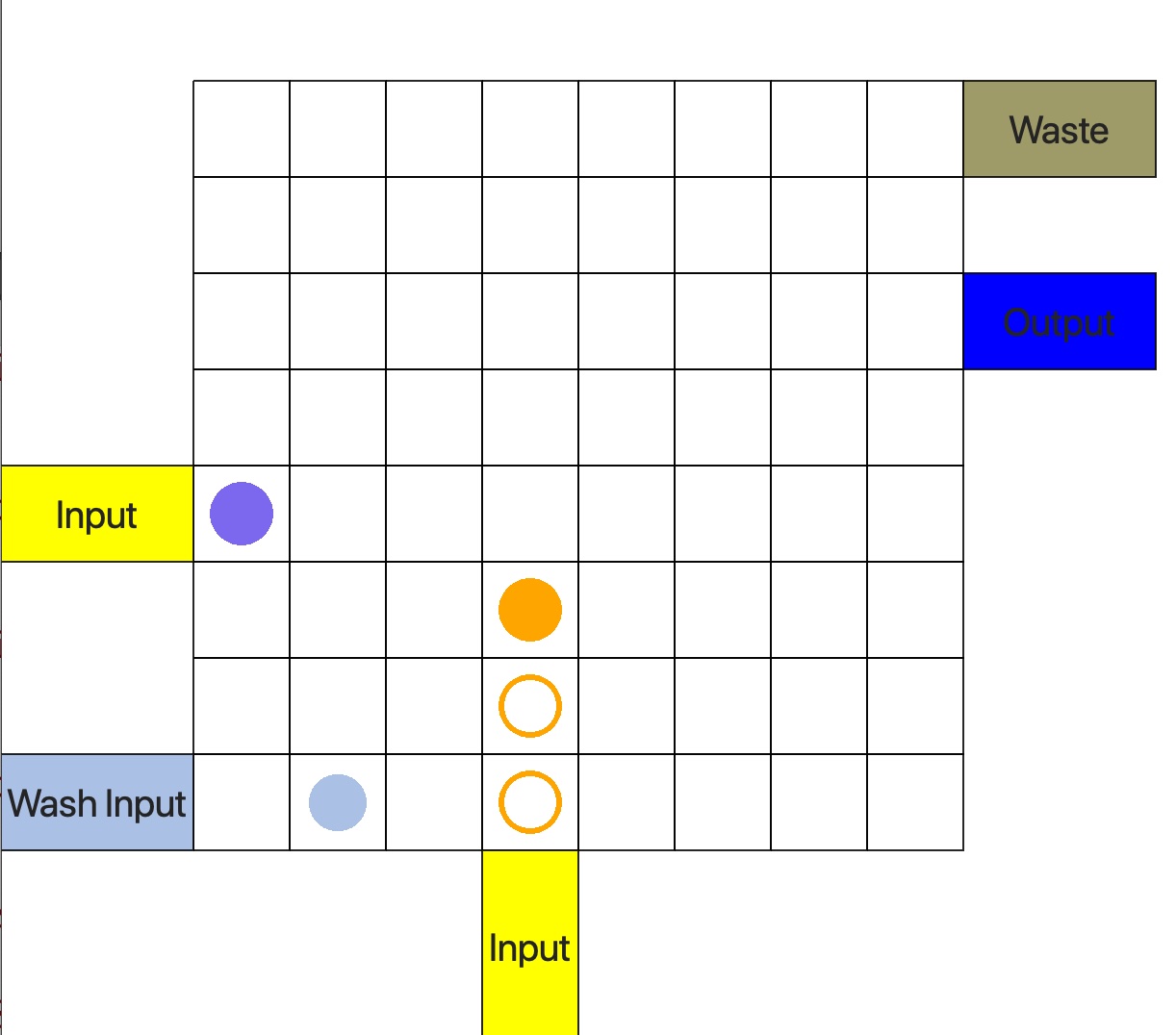
* 载入设置后立即显示初始的网格画面
* 工具栏中First, Prev, Next, Play分别实现复位、上一步、下一步、连续播放功能。用一个QLcdNumber组件显示当前时刻
* 鼠标移动到方格上时，可在状态栏中显示当前位置的坐标，如electrode (7,1)



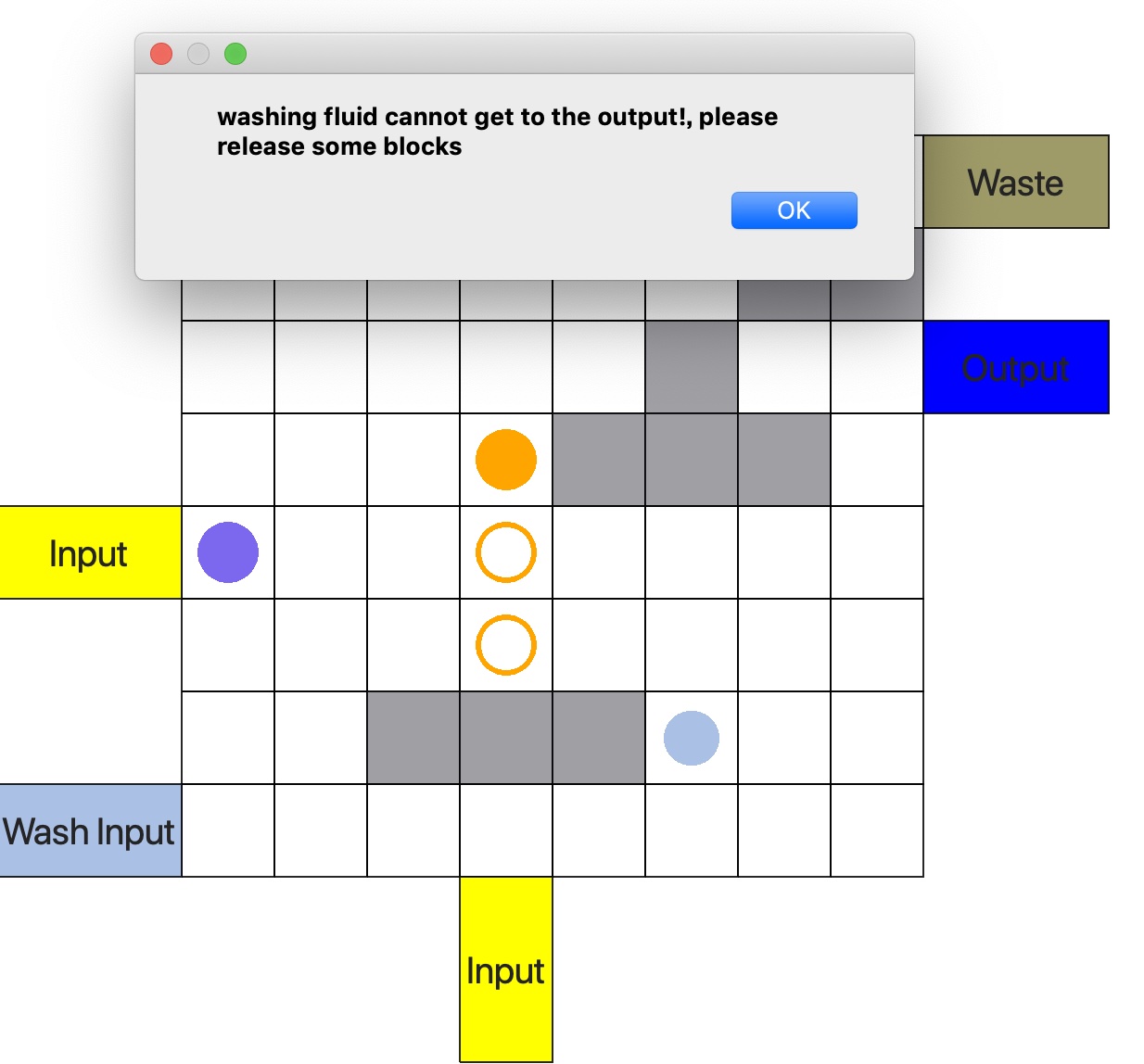
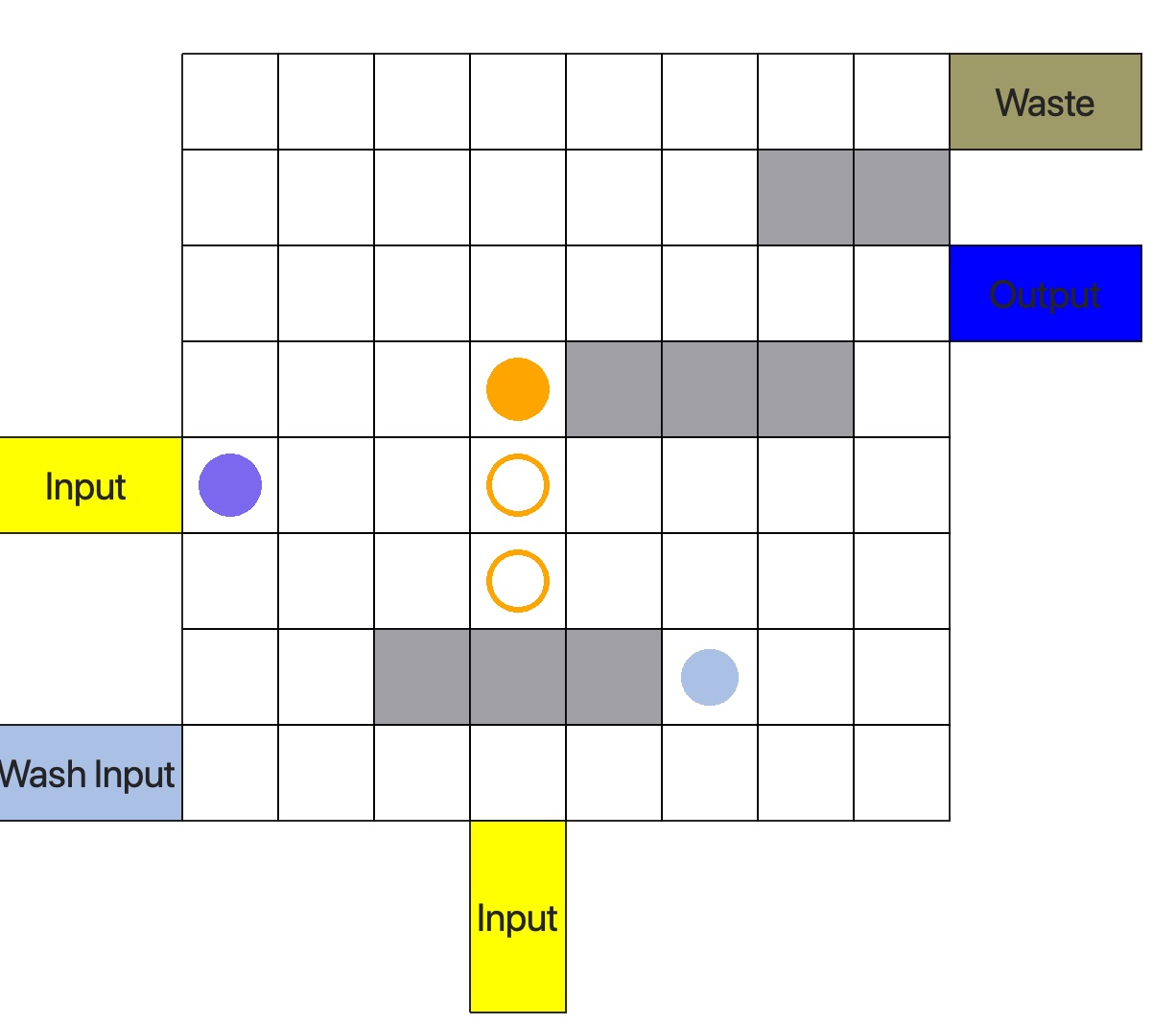
* 播放液滴画面时，用不同颜色表示不同的液滴，用空心圈表示液滴留下的轨迹。Merge和Split过后的液滴视作不同液滴，颜色改变。两个液滴Merge形成的新液滴体积增大。
* 连续播放时，每1s更新一次画面。
* 执行Move, Mix, Merge, Split操作时配有不同音效
* 点击Contamination按钮可以显示污染次数
* 当液滴违反静态约束或动态约束时会提示报错，同时停止运行
* 当Input和Output操作位置周围没有端口时报错。

1. 清洗模式

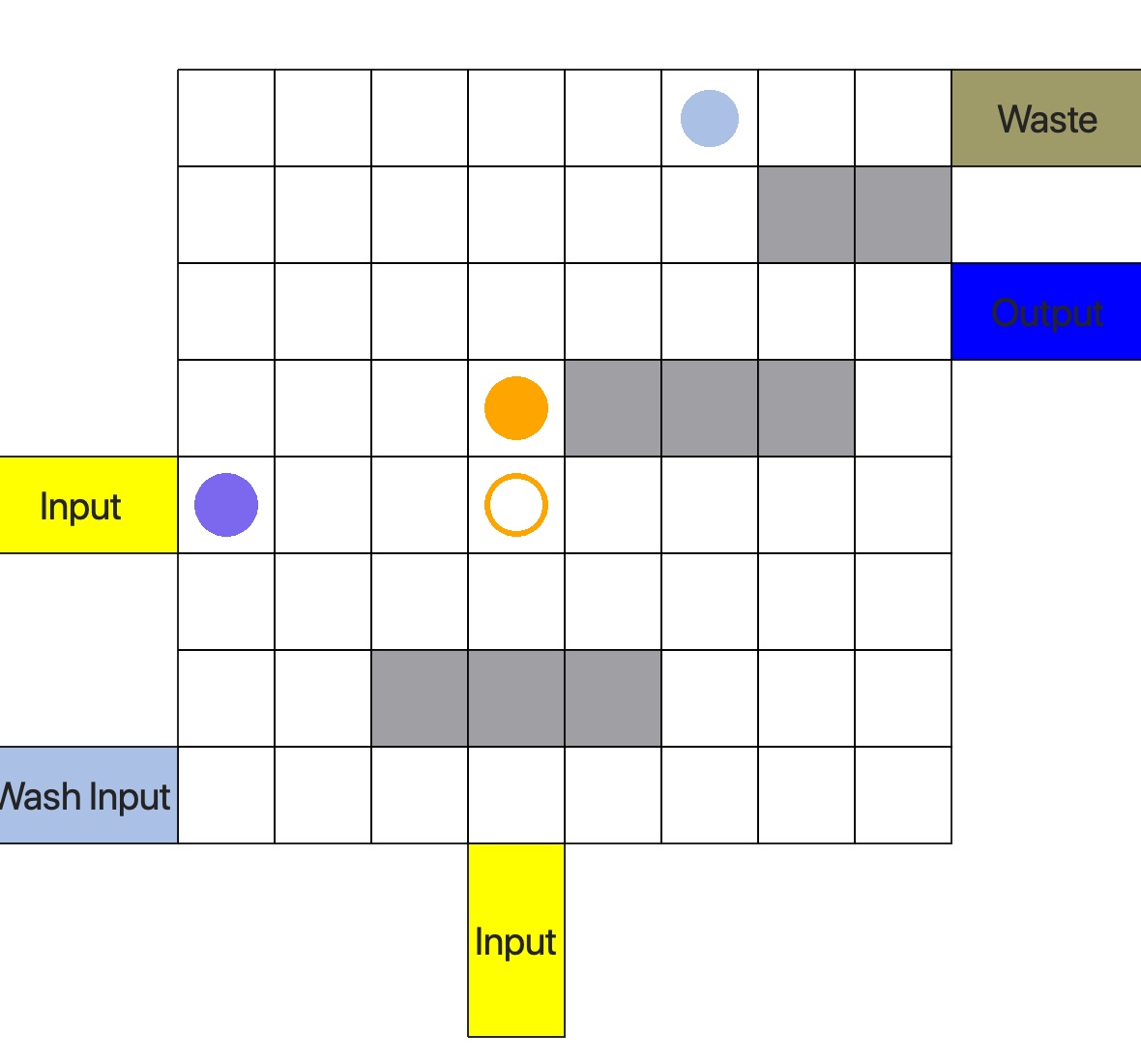
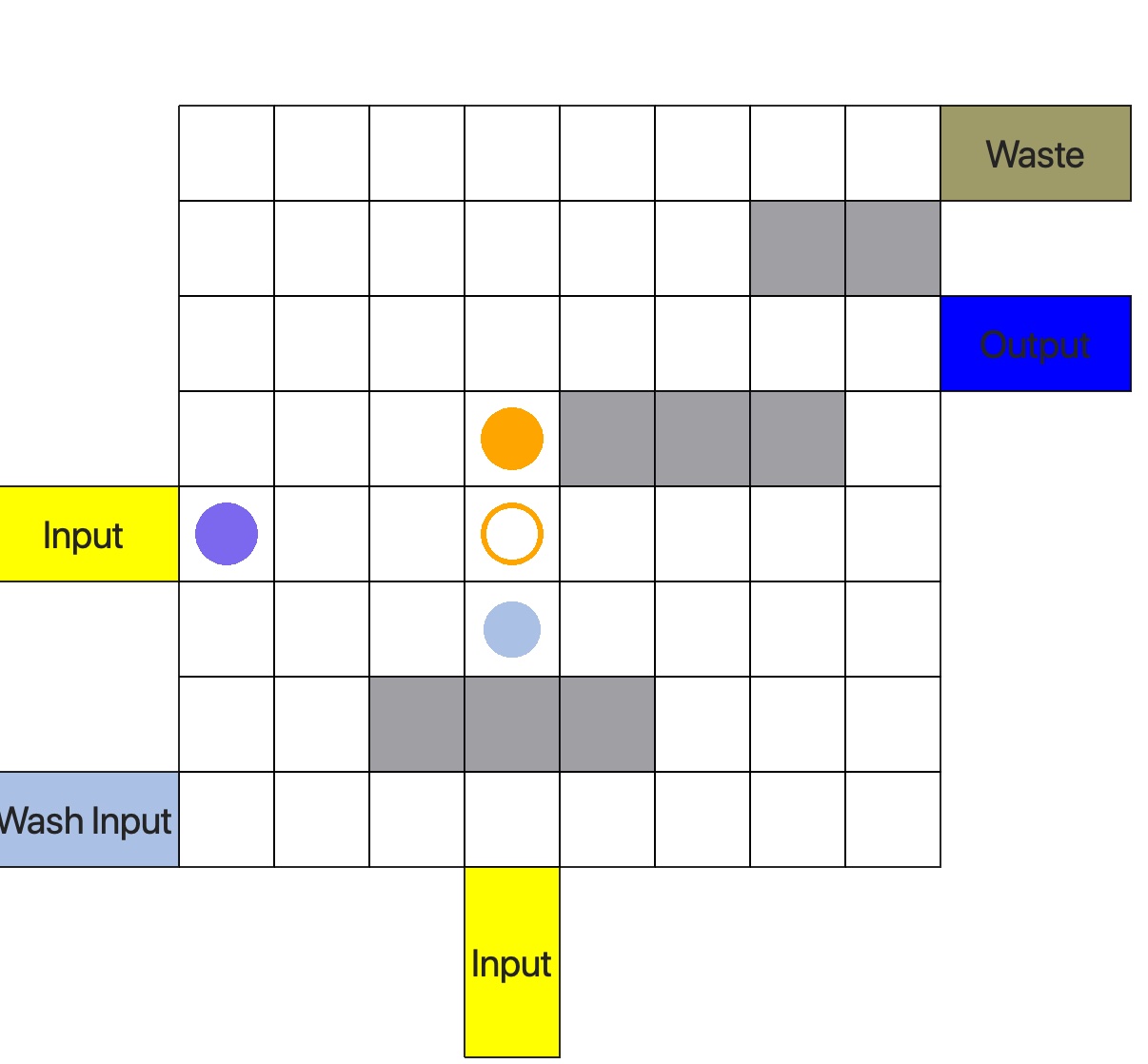
* 通过在开始窗口选中wash mode开启清洗模式，画面左下和右上会多出清洗的出入端口。在该模式下，用户只能使用Next操作，以及选择清洗液滴的障碍。



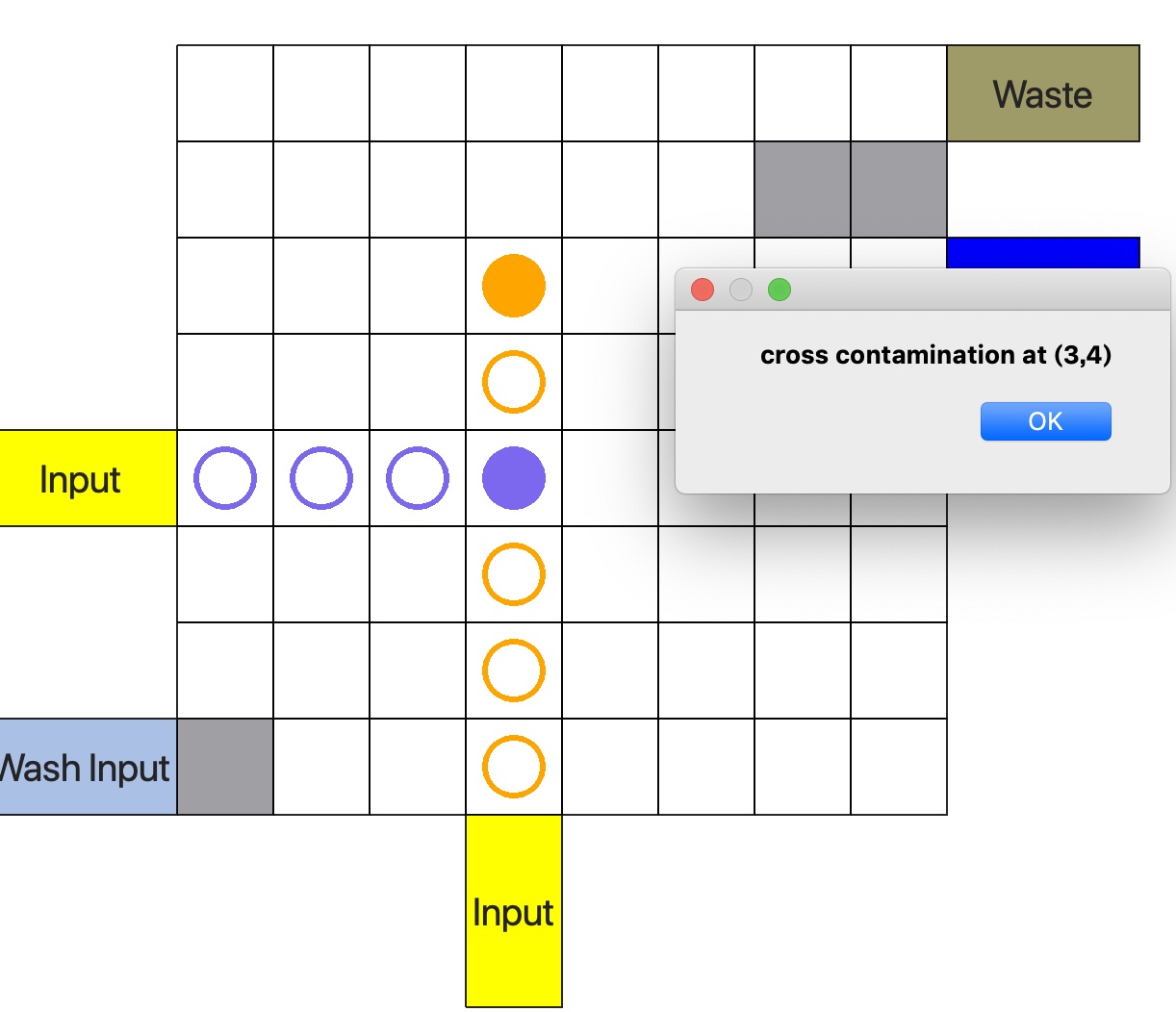
* 本程序采用的清洗策略是“尽力而为”，即网格中一旦出现清洗液滴可以处理（能够从输入端进入且能从输出端离开）的污染就派出清洗液滴进行处理。清洗液滴的路径满足静态约束，但在wash液滴清洗过程中其他液滴停止活动。
* 每个清洗液滴带着清洗一个特定位置的任务出发（中间可能也会清洗到别的污染），程序会计算出出发点到污染位置及从污染位置到出口的最短路径，交付给液滴执行。
* 清洗液滴支持动态规划路径，每当用户设置新的障碍，液滴都会清空已经计算好的路径重新规划，如果此时液滴发现无法到达输出端口将会弹窗报错。



* 当用户释放相应的位置时，当前在电极上的液滴将继续完成任务。如果发现没有可执行的清洗任务，会主动寻路到出口。



* 当出现两个液滴交叉感染，弹窗报错



1. **设计模式**
2. 整体架构

本程序使用QMainWindow的结构，将其CentralWidget提升为自定义的绘图类Graph。在mainToolBar中添加Action作为用户操作的入口。

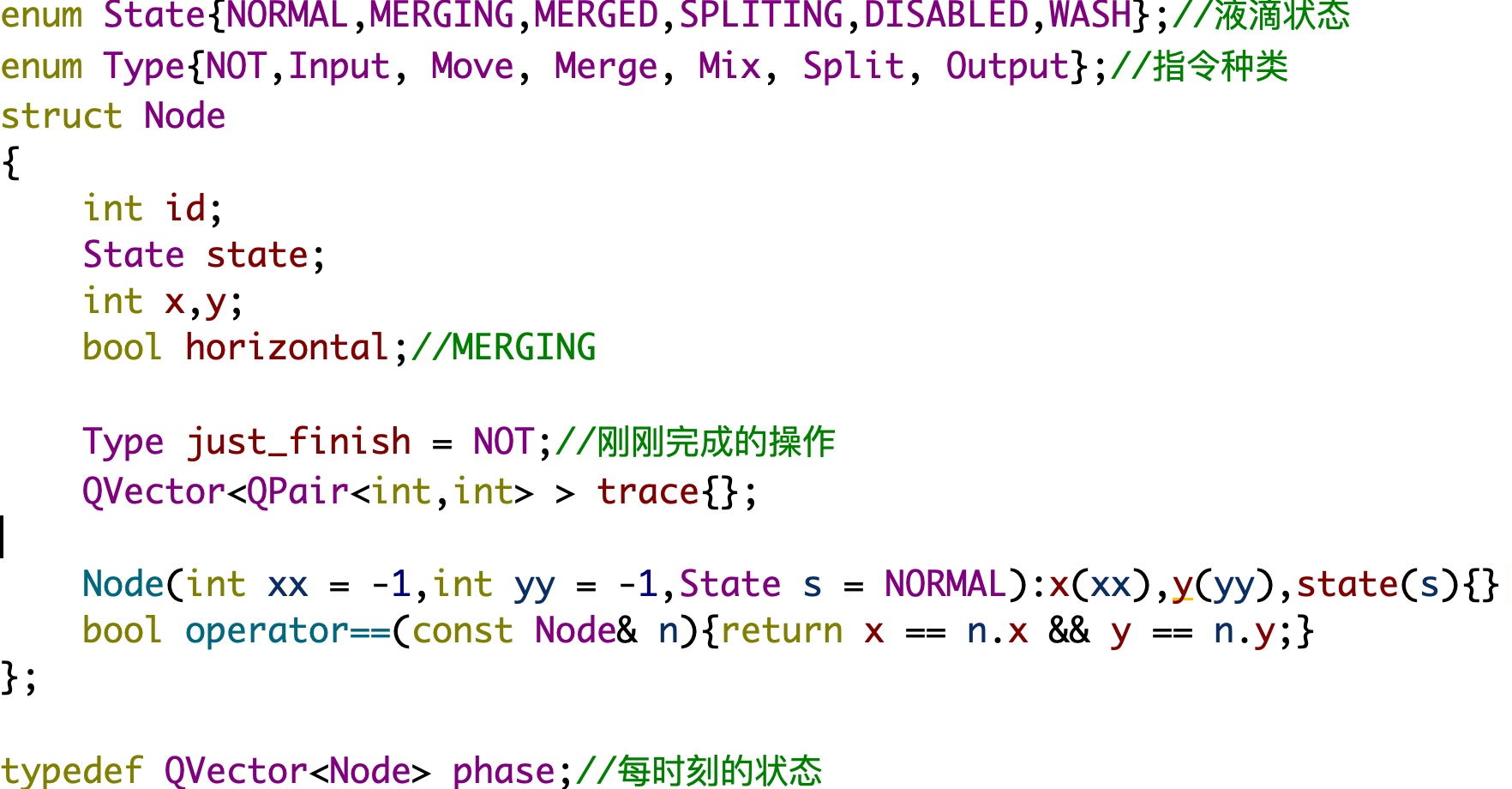
所有针对坐标的抽象数据处理封装在Core类中，所有功能的实现均调用Core类的接口，显示时由主程序向Core类请求一个状态(phase)并传给Graph类进行绘制。

1. 对话框

对话框负责初始化参数时数据是否合理，通过检验后将所有设置打包为Setting结构体，发射信号给主窗体，再分发给Core和Graph类进行初始化。

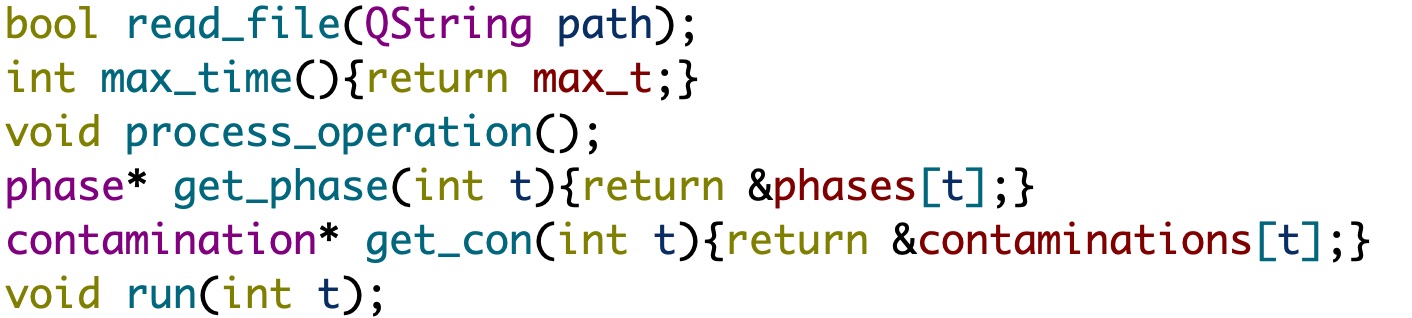
1. 数据存储

建立液滴类Node，在其中保存液滴的Id（液滴种类的标识），当前坐标(x,y), 当前的状态(正在合并/分离、普通、清洗、合并为大液滴)，刚刚完成的操作(用于播放音效)，液滴走过的轨迹等。将多个液滴的集合定义为phase，一个phase完整存储了运行过程中的一个时刻的状态，phase也是类之间相互传递数据的载体。



1. 操作执行、图形绘制、音乐播放

在非清洗模式下，所有的操作在读入文件后就都执行完毕并存储了。具体实现在Core::run(int t)中，最后将所有状态以及污染情况按时序存储在phases数组里，遇到时间请求直接返回相应时刻的状态。详见core.h, core.cpp。



在用户请求另一个状态时，主窗体会从Core中获取一个phase的指针传给Graph进行绘图。Graph类中有通过id分配颜色的函数，再根据节点中保存的各种信息绘制节点的状态和轨迹。Graph类同时承担的音频播放的任务，根据Node的just\_finish成员可以确定哪种声音需要播放。

1. 异常处理

在非清洗模式下，如果在执行过程中出现约束错误、出入口缺失错误，立即将最大时刻定为错误发生的时刻。同时在类中记录Error类型和错误位置信息，发送信号给主窗体，调用handle\_error()弹窗报错。具体的错误类型见error.h。

1. 清洗模式

清洗液滴的路径查找采用A-Star算法高效查找最短路径，实现在Wash::get\_path()函数中。清洗流程主算法为core::get\_wash\_phase()，其框架如下：

* 当前时刻无清洗液滴在网格上
  + 当前时刻有能够清洗的污染（可以到达且可以从出口离开）-> 派出新的液滴，并计算路径。
  + 当前时刻无可清洗污染 -> 执行原操作流程的下一时刻
* 当前时刻有清洗液滴在网格上
  + 当前液滴有未执行完的路径 ->继续按路径执行
  + 当前液滴路径为空（说明有用户改变约束条件，需要重新规划）
    - 当前有能够清洗的污染（可以到达且可以从出口离开）->从当前位置规路径并执行
    - 当前没有能清洗的污染，液滴规划走向出口的路径
      * 规划成功 -> 执行
      * 规划失败 -> 弹窗提示用户放宽约束条件