DMFB Simulator

计 86　周恩贤　2018011438

2019 年 8 月 24 日

1　程序介绍

本程序是基于 Qt 的数字微流控生物芯片模拟界面，允许使用者自定义芯片结构与操作指令，进而模拟实验过程。此程序提供播放功能，加入仿真音效；运行过程中，检测滴液的约束条件，在潜藏危机时提示使用者注意；实现了简易的清洗算法，尽量避免操作时出现滴液污染。可直接运行 *DMFB_Simulator.exe*，或使用 Qt Creater 编译运行 *DMFB_Simulator.pro*。

2　界面介绍

2.1　开始窗口

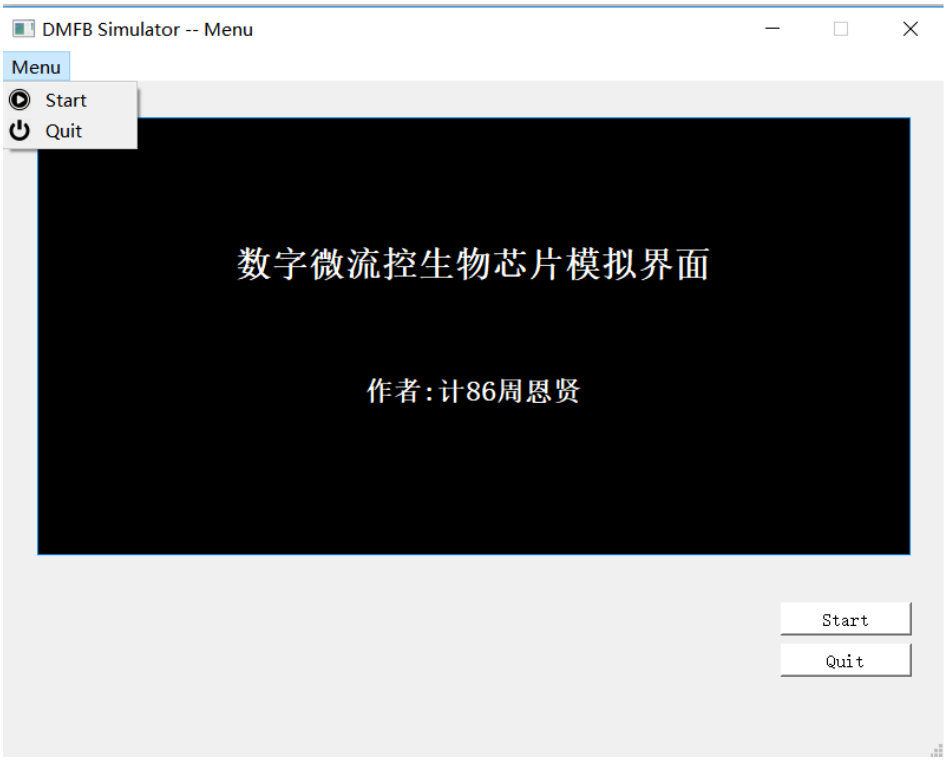

图 1: 开始窗口

图 1 是本程序的开始界面。点击菜单栏或右下角的 Start 按钮即可进入设置窗口。

2.2 设置窗口

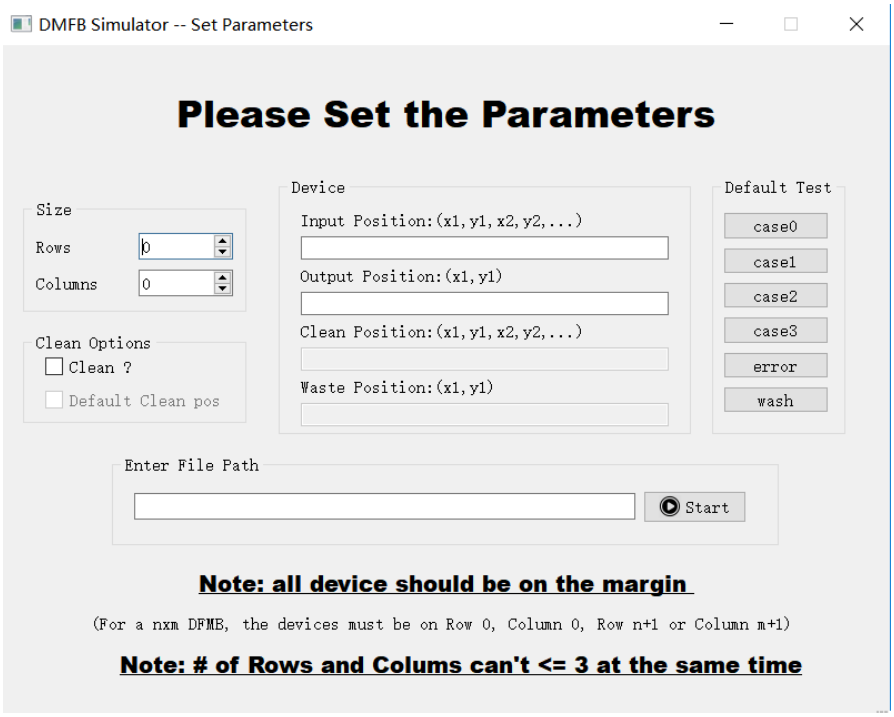

图 2: 设置窗口

图 2是芯片结构与操作文档的设置界面，使用者可按提示输入行数、列数、输入端口位置、输出端口位置、文档路径，并决定是否加入清洁功能。当选取加入清洁功能后，右方的文字框会**变亮可编辑**，允许使用者输入清洗端口、废液端口位置；若使用者使用预设的清洁装置位置，则文字框**变暗不可编辑**，并预设清洁端口在左下、废液端口在右上方。右方 `Default Test` 栏是内建的六个样例实验，点击后会在栏位中填入预设值，分别对应 `testcase1.txt,...,textcasewash.txt`，方便使用者快速进行模拟实验。

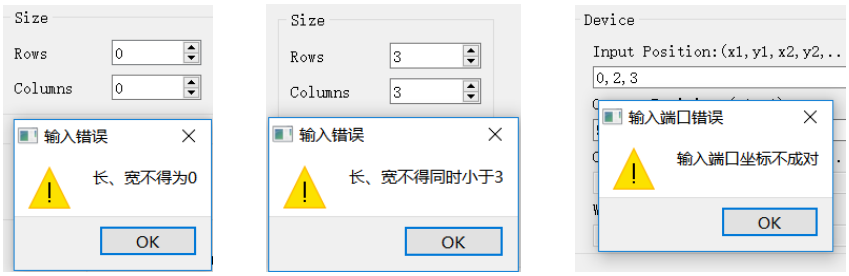

图 3: 错误检查

图 3 是点击 `Start` 按钮后**进行错误检查**的演示。若输入尺寸不符、坐标不成对、端口位置不符等等，会弹出提示窗提示使用者错误信息；若输入正确钮则进入主窗口。

2.3 主窗口

图 4 为模拟实验运行的主窗口。窗口的左半部会依照使用者刚刚所设定的值绘制芯片初始结构；同时，程序已依照文档路径进行读档，并剖析出所有时刻水滴的状态即芯片的模拟图。用户可按照右方的指示与相应的按钮操作。

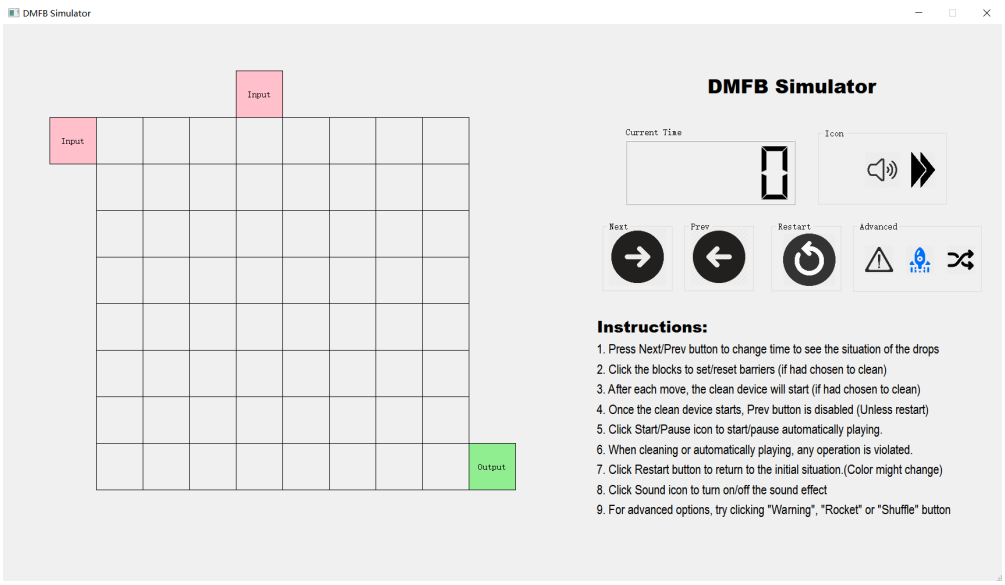

图 4: 主窗口

3 功能介绍

3.1 滴液操作模拟

依照文档指令，播放时滴液会进行不同操作。

3.1.1 Input、Output 指令

`Input` 指令：检查指令位置附近是否有输入端口，有则在指定时间生成随机滴液，否则报错。`Output` 指令与 `Input` 相似，同样会进行位置检查，并在指定时间排出指定滴液。

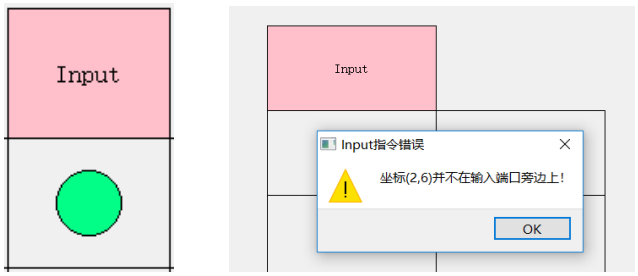

图 5: Input 指令

3.1.2 Move 指令

`Move` 指令：将指定滴液在指定时间移动一格，并在该电极留下污染。**污染进行了透明化并标示出留下污染的水滴编号**，同个滴液经过不会被污染，但若有其他种类滴液经过则会产生滴液污染。此时在绘图显示上由后者覆盖前者，但实际上**电极保存的污染可叠加，若未清洗而再次经过，则依旧会产生滴液污染**。

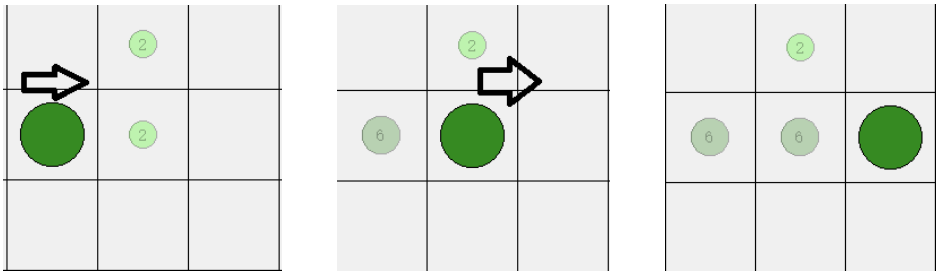

图 6: Move 指令与污染

3.1.3 Merge、Split 指令

Merge 指令：将两个指滴液在指定时间开始进行合并。下一时刻液滴被拉伸，融合成新的液滴；而再下一时刻两者合并成一个大液滴。Split 指令和 Merge 类似，将一个液滴分成两个相异的小液滴。Merge、Split 指令所产生的液滴都视为不同的，在指令完成时，三个位置**会分别留下三种不同的污染**。

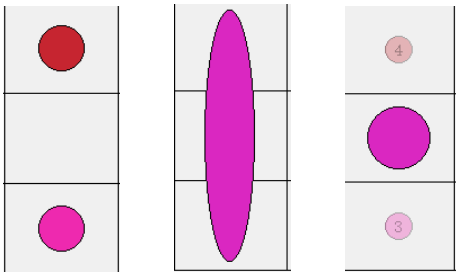

图 7: Merge 指令

3.1.4 Mix 指令

多步（通常为二、四、六步）移动指定液滴，通常用于加速 Merge 后的滴液。在本程序中**将 Mix 指令分解为多步 Move 指令实现**

3.2 操作界面

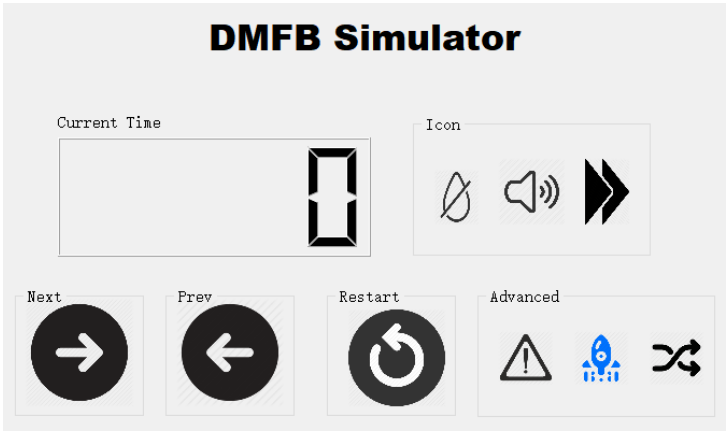

图 8: 操作界面

图 8 为可触发的操作界面，位于主窗口（图 4）的右上方，而右下方有相应的指示。

3.2.1 播放

用户可藉由以下按钮进行播放操作：

- 下一步：点击 `Next` 按钮，则时刻 +1 并绘制新状态
- 上一步：点击 `Prev` 按钮，则时刻 -1 并绘制新状态
- 重播：点击 `Restart` 按钮，则时刻归零并绘制新状态

负责播放时间的 `lcdnumber` 设有边界条件，避免出现负时刻或超出指令运行时间的情况。当液滴移动结束后，再次点击下一步，则会分别在每个电极上显示污染次数。污染次数为经

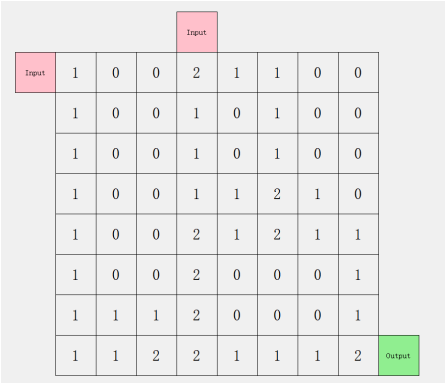

图 9: 污染次数统计

过该电极的**不同种**液滴个数。

3.2.2 自动播放

用户可点击图标进行自动播放。图 10 为触发自动播放时的界面，**单步播放、重播按钮消失**，而原本的图标转为暂停的图示。用户可点击暂停图示停止播放，此时暂停图标恢复成自动播放的图标，而单步播放与重播按钮再次出现。

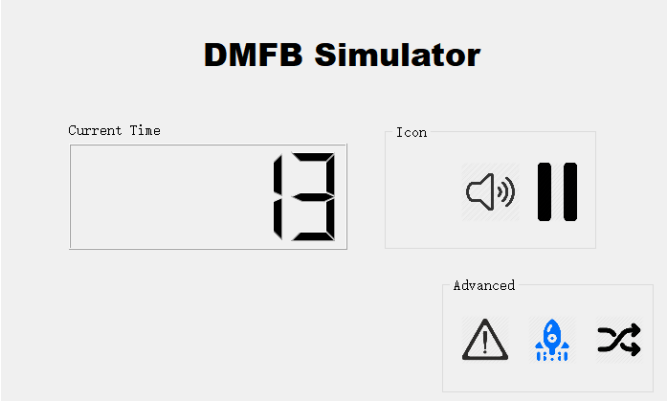

图 10: 触发自动播放，原按钮改为暂停图标，并隐藏播放功能

3.2.3 音效

图 11 为音效图标。在任何液滴移动、分裂拉伸、分裂成功、合成成功时，都会播放音效。点击图标可以设定是否开启音效。

图 11: 音效图标，点击会开启/关闭音效功能

3.3 约束检查

为了防止液滴在电极中移动产生意外的融合或污染，需进行约束检查：

- 静态约束：相同时刻的任意两液滴距离大于等于 2
- 动态约束：移动液滴在 t+1 时刻的位置与其他液滴 t 时刻位置的距离大于等于 2

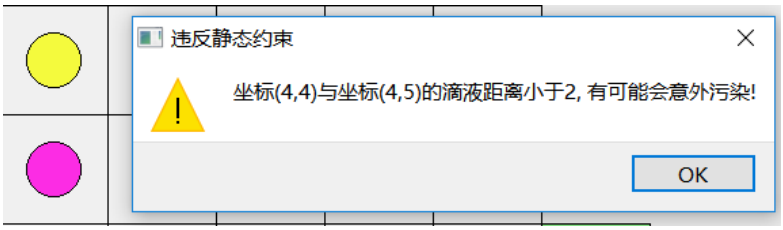

图 12: 违反约束条件的警告窗

图 12 为滴液移动违反约束时所跳出的警告窗。自动播放的过程中，**先进行目标时刻的动态约束检查，移动合法后再进行移动后的静态检查**。任一个水滴违反约束条件都会**立即停止**自动播放并弹出警告窗，同时停止滴液移动并回溯上一时刻，代表移动不合法。同时禁止下一步播放功能，直到点击上一步或是重播按钮。

3.4 自动清洗

为了防止滴液污染，我们实现了清洗功能。

3.4.1 界面设定

在设定窗口（图 2中）的 Clean Options 栏位，使用者可以决定是否加入清洗功能，并决定清洗槽、废液槽的位置。如果使用者选择加入清洗功能，则在主界面（图 4）的左半部会绘制出 Wash 端口与 Waste 端口。同时，在操作界面（图 8）中的 Icon 栏中会看到小水滴的图标，代表清洗功能尚未启动。

3.4.2 设置障碍

在静止（非自动请洗、非自动播放状态）时，如果加入了清洗功能，用户可以鼠标点击电极，该电极会触发变色，被变色的电极清洗液滴无法经过（不影响其他液滴）。同时，清洗液滴移动时也需要满足约束条件。

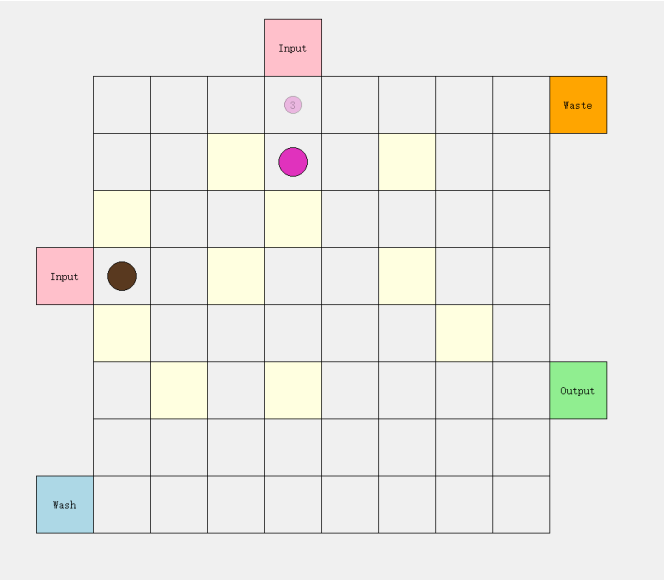

图 13: 鼠标点击，电极变色，设置/关闭障碍

3.4.3 开始清洗

当使用者按下一步（或是自动播放触发下一步）且液滴移动完毕后，会调用算法计算是否可清洗，若可清洗则进入清洗状态。清洗状态下**按钮区消失，无法进行其他操作**，且图标更改并给出"Patient, Cleaning Right Now" 的提示。被清洗液滴经过的地方污染会被带走，最后一同由废液槽排出。

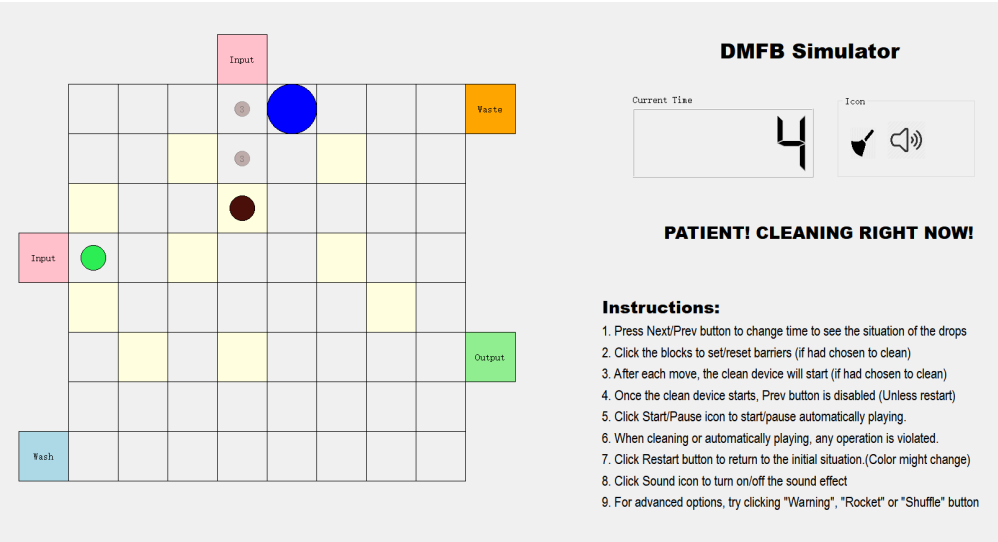

图 14: 清洗时的界面，蓝色大球为清洗液滴，右方出现清洗中的提示

清洗功能可视为不可逆（因为会随着每个时刻点击设置的障碍不同而有不同的清洗结果），故在清洗设备启动后，`Prev` 按键则会强制停用。只有当使用者按下 `Restart` 按钮从头开始执行时才会再次启用。

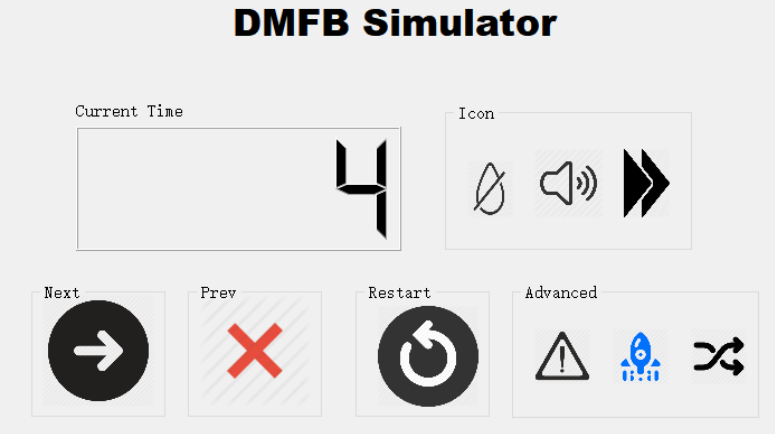

图 15: 清洗后的界面 (Prev 按键消失)

3.4.4 滴液污染判断

因鉴于约束条件、电极障碍等因素，并非清洗后都能避免掉滴液污染。若清洗完毕后下一时刻仍会产生滴液污染，则弹跳出警告窗告知使用者。

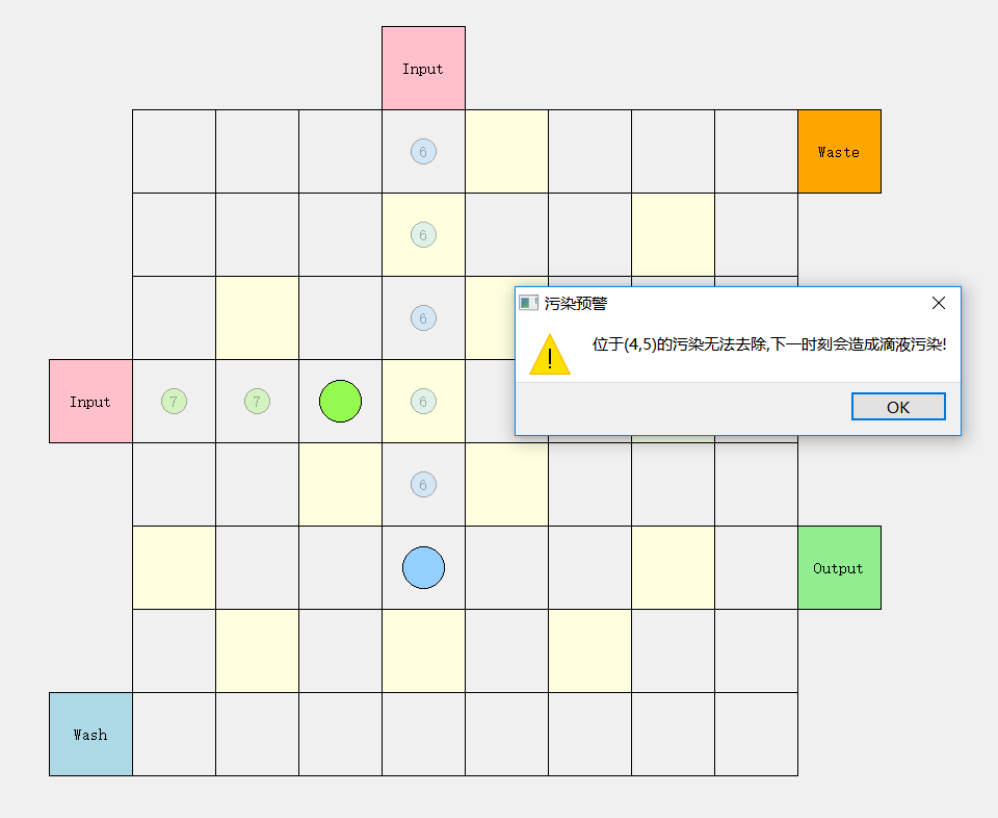

图 16: 滴液污染提示窗

4 Qt 界面实现

依照不同功能，调用了许多 Qt 库与函数方法：

- 音效：QSound
- 自动播放、清洗的延时：QTimer
- 绘图：写于 PaintEvent
- 设障碍：写于 MousePressEvent
- 图式化按钮：设定 stylesheet 中 border-image 的 URL
- 菜单栏：实现接收 Action 触发信号的槽函数
- 让物件消失/浮现：setVisible() 函数

由于许多图标、标签都是在 Qt Designer 中实现，无法使用数组进行管理。原本的方法是"Ctrl C + Ctrl V" 膜改大法，代码非常冗余。后来尝试后发现了一个方法：物件皆继承 QWidget 类，故可以**利用多态的方式使用 QWidget* 的数组**进行同步管理。

```
QWidget* movelist[11] = {ui->nextBox, ui->nextButton, ui->prevBox,ui->prevButton,ui->restartBox, ui->restartButton, ui->autoButton,
                         ui->advanceBox, ui->proButton, ui->warningButton, ui->shuffleButton};
for (int i = 0; i < 11; i++)
    movelist[i]->setEnabled(!a);
```

图 17: 用 QWidget* 数组同步控制 ui 物件，简化代码

5 编程与算法实现

5.1 处理水滴

我写了一个水滴类 drop 来负责管理单个水滴。水滴的私有属性 type 包含：空白、正常水滴、水平拉伸、垂直拉伸、拉伸旁的小点、污染六种。同时，在正常水滴生成时会有独一无二的编号 index，并随机分配一组 RGB 值与预设大小。合并、分裂时产生新编号与颜色，同时按照等面积的方式设置大小（也就是说，**两个同样大小的水滴合并后 size 会变为 $\sqrt{2}$ 倍**）。

5.2 处理绘图

为了能正确画出每一时刻每一电极上的正确状态，我写了一个表格 table 类代表一个芯片。视窗中有一个 <vector<table>> tablelist 分别管理每一时刻的芯片，而每一个芯片（table）上有 vector<drop> droplist 管理每一个电极上的水滴状态。因此，第 t 时刻的编号 i 电极上的水滴就可以表示为 tablelist[t].droplist[i]。在 PaintEvent 函数中，依照时间遍历所有电极上的水滴即可完成绘制液滴的功能。

5.3 处理污染

由于同个液滴重复经过仅算一次污染，不可重，我的第一个反应就是集合。因此，使用 vector<set<int>> counter[MAXTIME] 管理每一时刻、每一电极上面的污染编号集合。因此，t 时刻编号 i 电极上的污染集合就可以表示为 counter[t][i]，每有新的液滴经过就用 insert() 加进集合中、判断是否造成滴液污染用 find() 遍历、最后使用 size() 显示次数。

5.4 清洗算法

在基础功能中，由于清洗时可以" 时停" 且不考虑清洗时长与清洗液滴的容量，故我的方法，也是很贪心很暴力的方法，" 能洗则洗"。每一次时间改变后利用 BFS 遍历所有合法的污染点，并连出一条通路，最后连到废液口排出。若没有可清扫的污染点或是无法到达废液口，则不清扫。这个方法简单也看似正确，在多数情况下运行结果也不错，但是却存在着至少三个问题：

- 耗时且耗容量：清洗的真正作用在于" 避免液滴污染"，真正只需要清洗的污染点其实只有多个液滴重复经过的地方，而不需要大范围的全部清扫
- 机动性差：只要清洁口/废液口被封住了就无法进行清洗
- 误判情形：假如在 t 时刻废液口被封住了，但可以清扫到关键性的污染点。一个更好的方式或许是先清洗后静置清洗液滴，随着时间变化移动清洗液滴以避免影响到其他滴液（同时满足所有约束条件），待到废液口开启（比如 t+1 秒时）时再视情况排出。如果指按照当前时刻采取暴搜方法，很有可能就错过唯一能清扫到污染的关键时刻了

不过，因为时间紧迫以及算法能力仍有待加强，我并没有实现其他更好的清洗方案。希望在未来能补强算法能力，写出一个更有效省时的解决方案！

6 附加与其他

由于算法不是很精熟，也还没想好该如何同时处理播放清洗液滴与其他液滴的交互方式，因而附加功能都未实现。但作者还是很贴心的在 Advanced 栏中新增了三个按钮，并提示使用者可以点击看看。

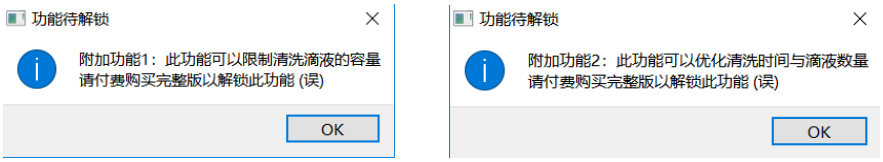

图 18: 点击进阶功能所弹出的消息框

图 18 为点击进阶按钮后会弹出的消息框，告诉使用者该升级完整程序版程序才可解锁此功能~~（然而真实情况是还没写出来）~~。希望有朝一日完整版程序能够顺利完成！

虽然没有完成指定的附加功能，但我也尝试在各种方面去优化程序：

- 实现了图标点击切换图片的对应功能
- 实现了自动播放的暂停功能
- 提供使用者自定义的清洗槽入口、废液槽入口
- 实现带清洁的自动播放功能
- 图标化触击按钮，美观且简洁
- 依照提供的样例，设置预设实验以方便进行调试

- 完善各功能间的逻辑关系，并以图形化界面明确的告知使用者，如：
  - 设定窗口中，清洁框、预设清洁位置框以及自定义输入清洁端口位置的逻辑关系
  - 自动播放时禁止其余播放操作，原本的图标改为暂停按钮
  - 自动清洁时按纽消失，并显示" 清洁中" 的指示
  - 清洁后不允许回溯，重启后才允许

虽然这些都是小功能，不过我在实现这些功能的时后，也是在尝试以使用者的角度去换位思考：想想使用者可能会进行什么操作，避免多个功能同时触发，或是产生误触按钮的情形；同时，尽量让界面保持美观简洁，好用易上手。

7 心得

第一次接触 Qt，十分新鲜有趣~~（爆肝煎熬）~~。要在如此短暂的一星期内，完整的学习并精通 Qt 是不太可能的。但在小学期的第一周，我就深深体会到了**查询资料**与**实际操作**两件事情的重要性。

首先是查询资料，因为 Qt 多数基础物件都是封装好的，关键在于如何看懂工程文档，正确的使用接口，并留意所有细节。比如说我一开始没注意看说明，调用了父物件的 `setEnabled(false)` 函数,结果才发现所有子物件都无法触发了。再次查询文档才发现其中明确的写着：`"Disabling a widget implicitly disables all its children"`，只能说魔鬼藏于细节之中！

再来是动手操作，因为很多时后光去想不去尝试是没有用的。一开始写小作业的窗口都非常的简陋，但经过多次练习后，现在至少已经掌握一些使用 Qt Designer 的方式与美观界面的方法了。换句话说，要勇敢尝试，熟能生巧！

这次大作业的附加功能没有实现，非常可惜，但也意味着我在算法方面还有很多进步空间，未来要继续加油！