程序设计训练 第1周大作业报告 ——数字微流控生物芯片模拟界面

计86 2017011998 冯卓尔

设计背景

微流控生物芯片是把生物、化学、医学分析过程的样品制备、反应、分离、检测等基本操作单元集成到一块微米尺度的芯片上，自动完成分析全过程。本次大作业的目的，是模拟在微流控生物芯片上的对象发生的动作，以及实现虚拟的芯片大小设计与动态模拟，为实际生产研究提供理论化模型，减少研究遇到的困难。

程序运行

本项目所使用的是QT GUI框架，它能够提供便捷的UI绘制，同时利用信号-槽机制实现高效的交互。运行时，QT的版本应当高于5.7。使用前，先删去外部的build-dmfb_asaharu-Desktop_Qt_5_13_0_clang_64bit-Debug文件夹，并且删去当前目录下的dmfb_asaharu.pro.user，防止不同版本编译引发的错误。使用QT Creator打开dmfb_asaharu.pro，选择clang作为kit进行编译，能够运行本项目。注意，由于开发操作系统为macOS，因而内部代码中会使用系统适配的API。

框架设计

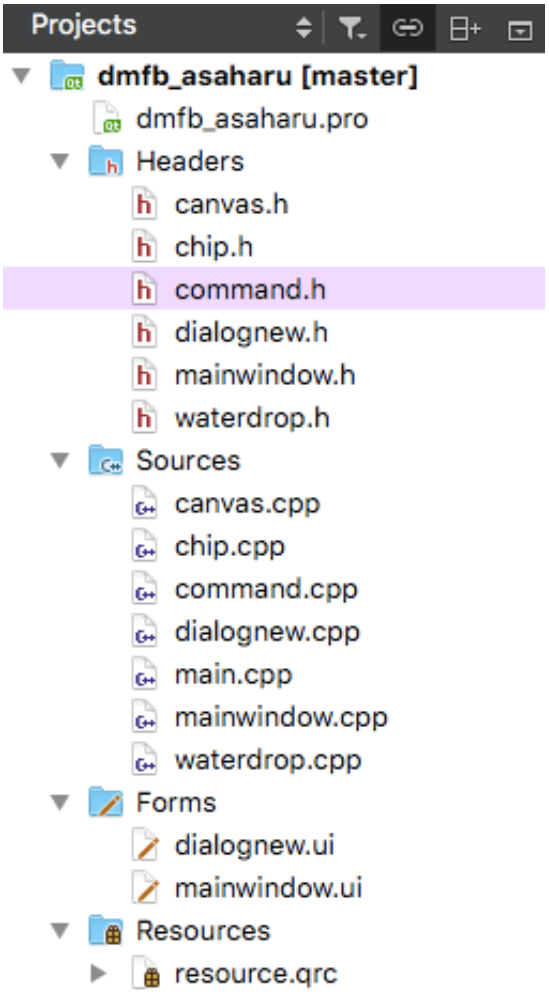

图1 项目框架

本项目基于QMainWindow进行设计，工程项目中增添了五个头文件，分别是`chip.h` `command.h` `dialognew.h` `canvas,h` `waterdrop.h` 其中，`dialognew.h` 是创建输入芯片大小以及各个水滴通道位置的对话框创建类，`canva.h`主要负责图形界面的绘制，`chip.h`中含有两个类，一个是Cell，抽象芯片中的格子（Grid），另一个是Chip，用于抽象芯片。另外，由于设计txt文件读入，设计师创建了`command.h`文件，里面有主要负责记录命令内容的Command类，以及将整个命令读取与执行统一封装的CommandFactory类。除`dialog.h`相对独立，其他类之间的关系由下面的UML图给出。

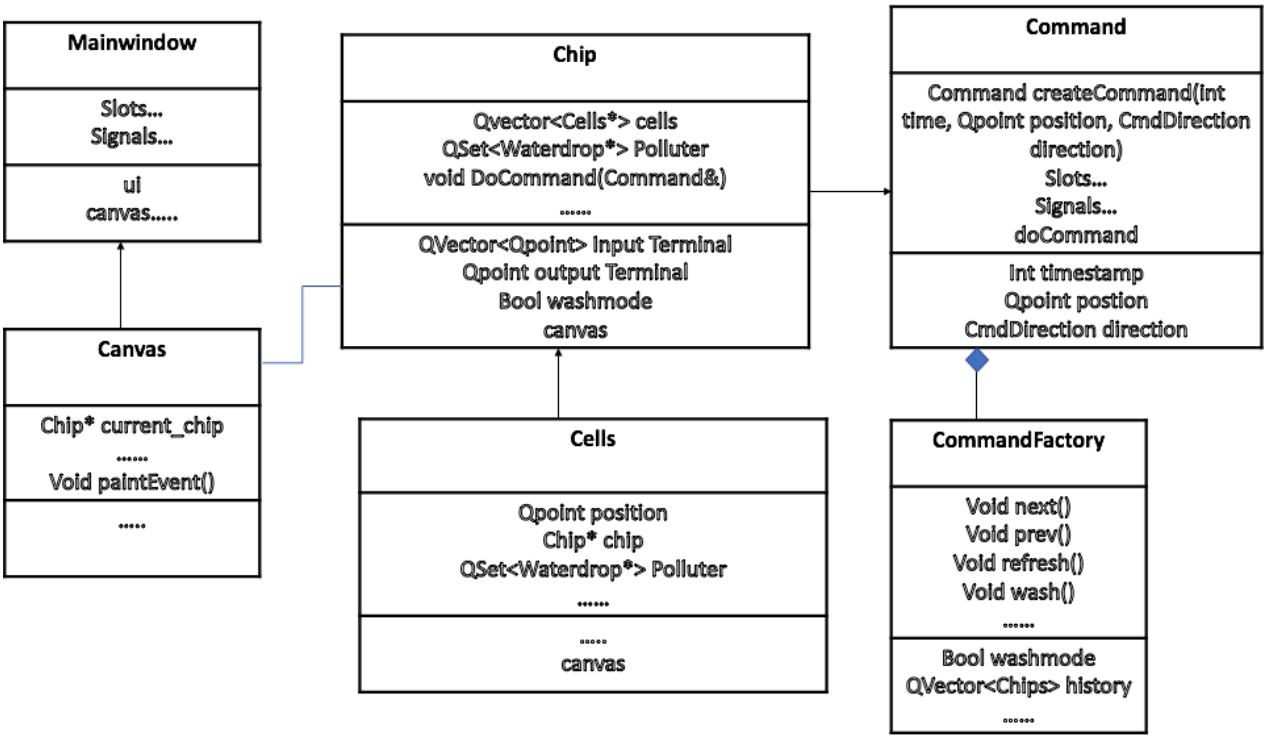

图2 类间关系及主要成员UML

功能说明

根据需求文档，开发者实现了芯片模拟界面以下功能

初始化芯片大小与输入输出端

通过菜单栏或工具栏按钮，弹出窗口。用户通过该弹出窗口输入DMFB的行数和列数，以及输入端口（可以有多个）、输出端口（1个）的位置。点击确认按钮后，在界面绘制芯片的初始结构。

其中，只能通过输入端口将液滴输入到端口相邻的电极上，只能通过输出端口将端口相邻电极上的液滴输出。

另外，程序能够实现错误检查：行数和列数不能同时小于等于3；输入端口应当在芯片的边界上；输出端口应当在芯片的边界上。具体的界面如下图，在每行中输入正确的数字，点击OK按钮就可以生成芯片。Input Check是用于显示错误的标签，如果有非法输入，系统会判断错误类型并且在此用红字显示。

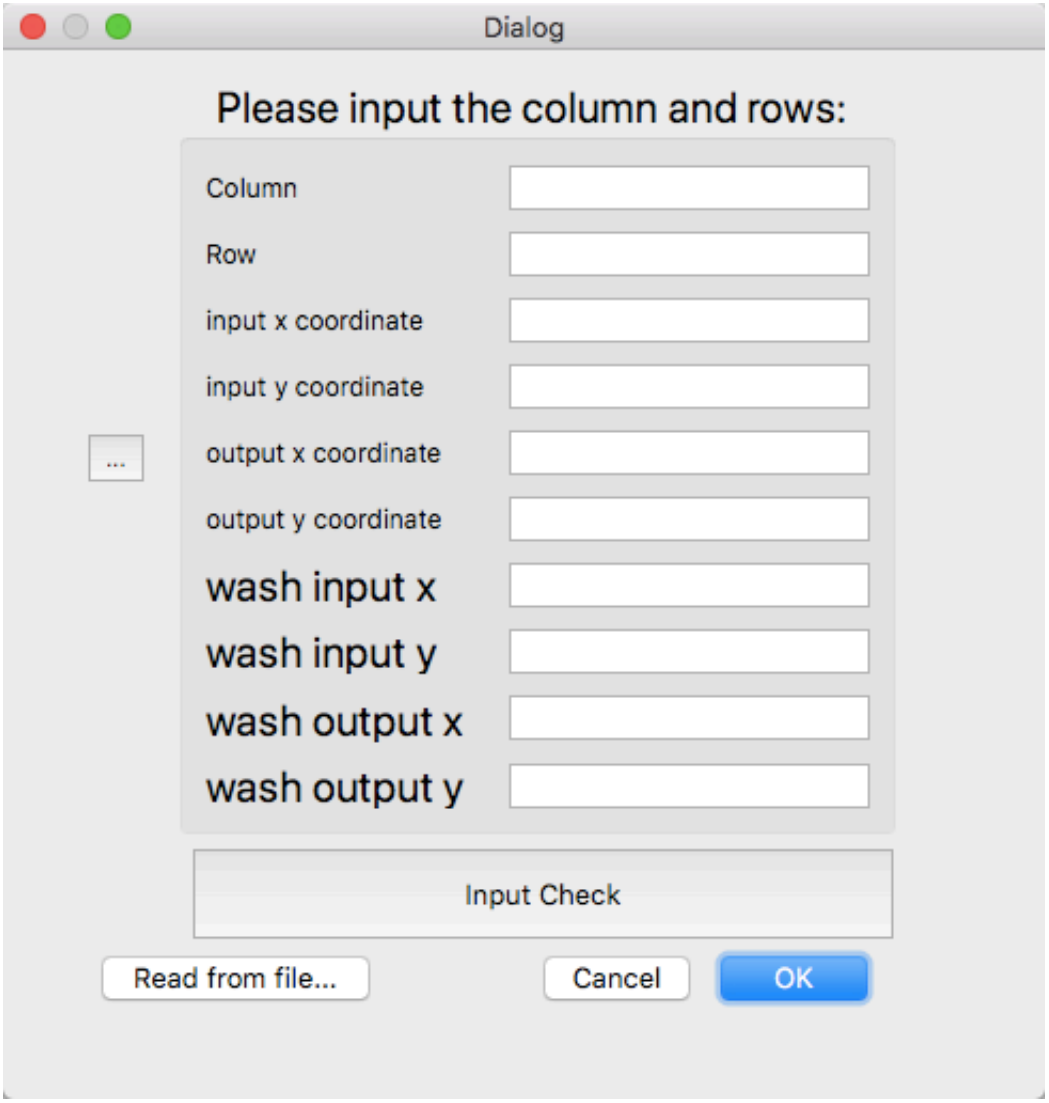
Dialog
Please input the column and rows:
Column
Row
input x coordinate
input y coordinate
...
output x coordinate
output y coordinate
wash input x
wash input y
wash output x
wash output y
Input Check
Read from file...
Cancel
OK

图3 初始化界面对话框

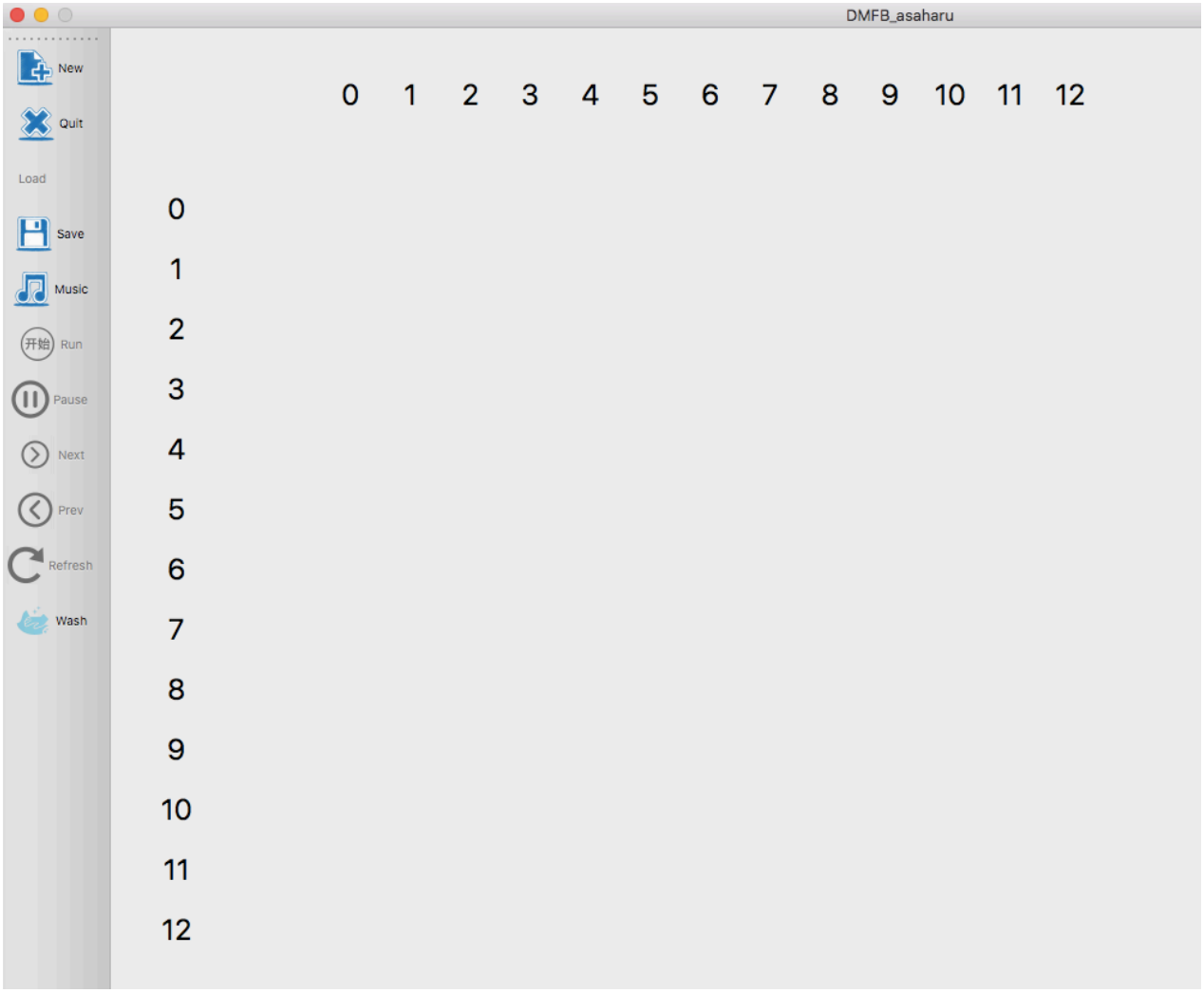

图4 未初始化的芯片界面

动画播放

程序实现了移动过程的单步播放与连续播放：

添加“下一步”与“上一步”按钮，可以按照时间，单步展示每个时刻芯片上的液滴状态。

添加“全部播放”按钮，从当前时刻开始连续播放芯片上的液滴状态，直到结束。

添加“复位”按钮，返回时刻为0时的液滴状态。

其中，Next为后一步按钮，prev为前一步按钮，refresh为重新播放按钮，开始为开始播放按钮。当芯片被正确初始化并且读入了正确的命令文件后，芯片的Run按钮变为可点击。在点击Run按钮之后，Refresh按钮变得可以点击。在此期间，next和prev按钮会根据当前chip的时间戳timestamp来判断是否能够上一步/下一步并且实现相应功能。

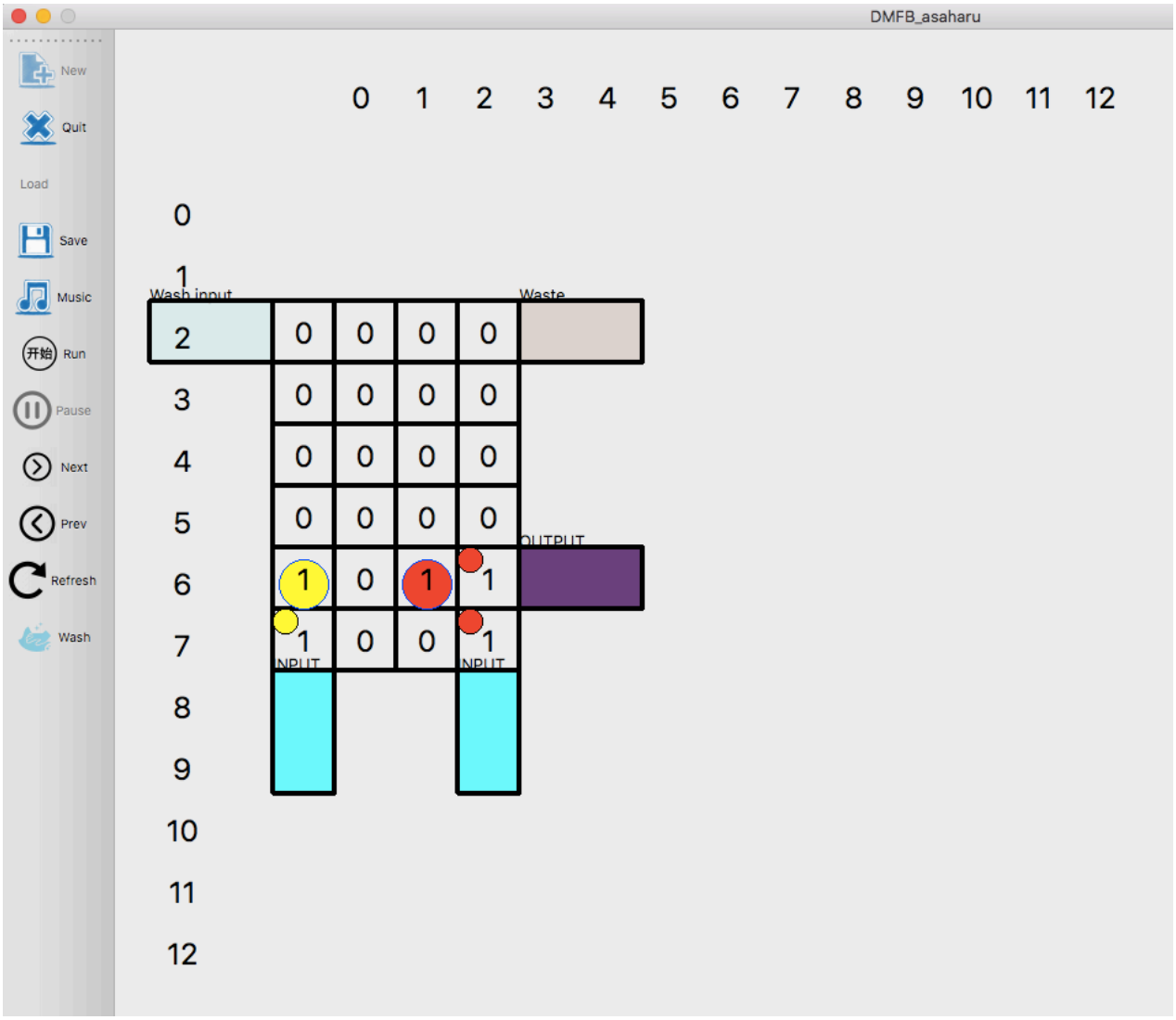

图5 播放动画

canvas.h中canvas类的代码

```
// canvas.h
class Canvas : public QWidget
{
    Q_OBJECT
public:
    Canvas(QWidget* parent = nullptr);
    ~Canvas();
    bool should_paint = true;

    void paintEvent(QPaintEvent *ev);
    void keyPressEvent(QKeyEvent *ev);
    void mousePressEvent(QMouseEvent *ev);
    void mouseReleaseEvent(QMouseEvent *ev);
    void mouseMoveEvent(QMouseEvent *ev);

    void drawChip();
    void drawGrid(QPainter&);
    void drawTerminal(QPainter&, QPoint, QColor, QString);
```

```
    void drawCleanWaterTerminal(QPainter&);
    void drawWaterDrop(Waterdrop*);
    void drawWaste(Cell*);

    bool getCleanMode();
    void setCleanMode(bool checked);

    void initCommand(QString& info);

    void pause();
    void play();

    bool isEnd();

    Chip* my_chip;
    CommandFactory* get_command_runner();
    QTimer *timer = nullptr;
    ///bfs
    void setWashCommands(Chip* initial_chip);
    QMap<int, QVector<Command>> wash_commands;

signals:
    void play_over();
    void music_begin();

public slots:
    void init(int, int, QVector<int>, QVector<int>, int, int, int, int, int,
int);

private:
    int column = 0;
    int row = 0;
    int grid_height;
    int grid_width;
    bool clean_mode = false;
    bool initialized = false;

    CommandFactory* command_runner;

    QPoint start_point;

    QVector<QPoint> Input;
    QPoint Output;

    QPoint wash_input;
    QPoint waste_output;

    QVector<Command> command;
```

```
    QVector<Chip*> AllChips;

    int current_timestamp = 0;
    int wash_timestamp = 0;

};
```

播放音效

程序实现了液滴移动时的音效功能：移动(Move)与混合(Mix)应当具有相同音效；分离(Split)时，液滴拉伸与最终分离应有不同音效；合并(Merge)时，合并成功时应当具有音效。

基于QMediaPlayer可以播放mp3音乐，基于QSound可以播放wav音乐。在课余时间通过雀魂平台研究立直麻将的时候发现了其音效短而贴切，因而开发中通过Chrome浏览器解析收到的网络请求文件获得相应的音效，并把他们加入资源文件中，最后在项目中使用。具体的实现位置在`chip.cpp` 200 -250行前后。

静态、动态约束检查

令(x_i^t, y_i^t)为t时刻的液滴D_i所在的位置，满足静态约束$|x_i^t - x_j^t| < 1$或$|y_i^t - y_i^t| < 1$，同时动态约束$|x_i^t - x_j^{t+1}| < 1$或$|y_i^t - y_j^{t+1}| < 1$.

在具体的实现中，在实现上一步、下一步的过程中，在chip类中引入了history这个QSet，因而我在每次执行一次move命令时，调用history查看下一步中芯片中所有液滴的位置，将动态约束和静态约束转化成为两次静态约束判断。具体代码在`command.cpp` 1 - 200行左右do_command()里进行了判断。

绘制液滴移动污染

程序实现了液滴移动时的污染情况绘制：液滴经过电极后，在电极上标注污染情况(标注方式自己确定)，一种液滴重复经过一个电极只记录一次污染，混合后的液滴认为是新的液滴，分裂出的液滴也认为是新的液滴。液滴移动结束后，在每个电极上显示污染次数。

如图，这是一个即将播放完成的动画，在动画中小圆表示污染，背景中的数字表示污染次数，大圆表示当前液滴，液滴与相应的污染为同一颜色。

该部分的具体实现是在Cell类中加入一个QSet进行记录，每次有水滴经过此处时，QSet会录入该水滴进行记录。另外Cell的current_drop也会被记录称为当前的水滴，prev_drop会在当前水滴离开后记录。这样子就实现了记录功能。具体代码在`command.cpp` 1 - 200行左右do_command()里进行了判断。

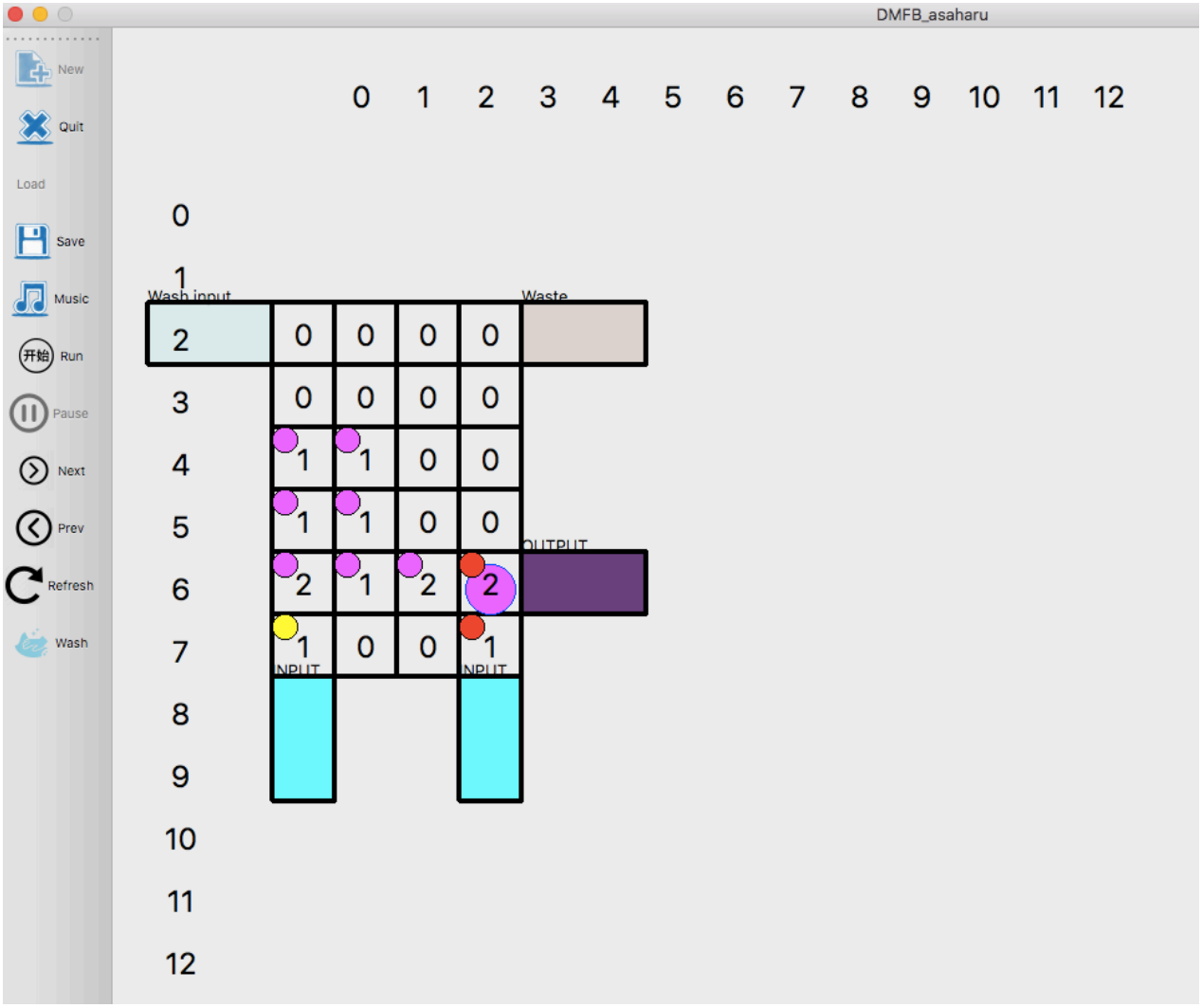

图6 污染液滴绘制

清洗芯片功能

程序引入了清洗液滴，实现了液滴移动时的路径规划，防止出现液滴污染。两个不同液滴先后经过同一个位置时，后者将被污染；默认清洗液滴入口(wash input)在芯片左下，废液回收端口(waste)在右上。也可以自行定制，但入口应不同于其他液滴入口；清洗液滴从wash input进入芯片，可以去除路过电极上的污染，最终从waste移出芯片。

如图为相应的实现，鼠标左键点击了相应网格之后，网格被设置成为了水滴不可经过的。由于水滴的运动默认为非常快，因而设计默认开启清晰模式之后，液滴每移动一步，都使用水滴对整一块芯片进行清洗，因而问题转化成为清洗的区域大小。图中浅蓝色的区域便是清洗液滴能够达到的区域，开启清晰模式之后，液滴每移动一次该蓝色区域都会被重新计算。

为了拓展清洗液滴类，我在Cell、chip类中添加了Washmode来判断当前是否为清洗模式

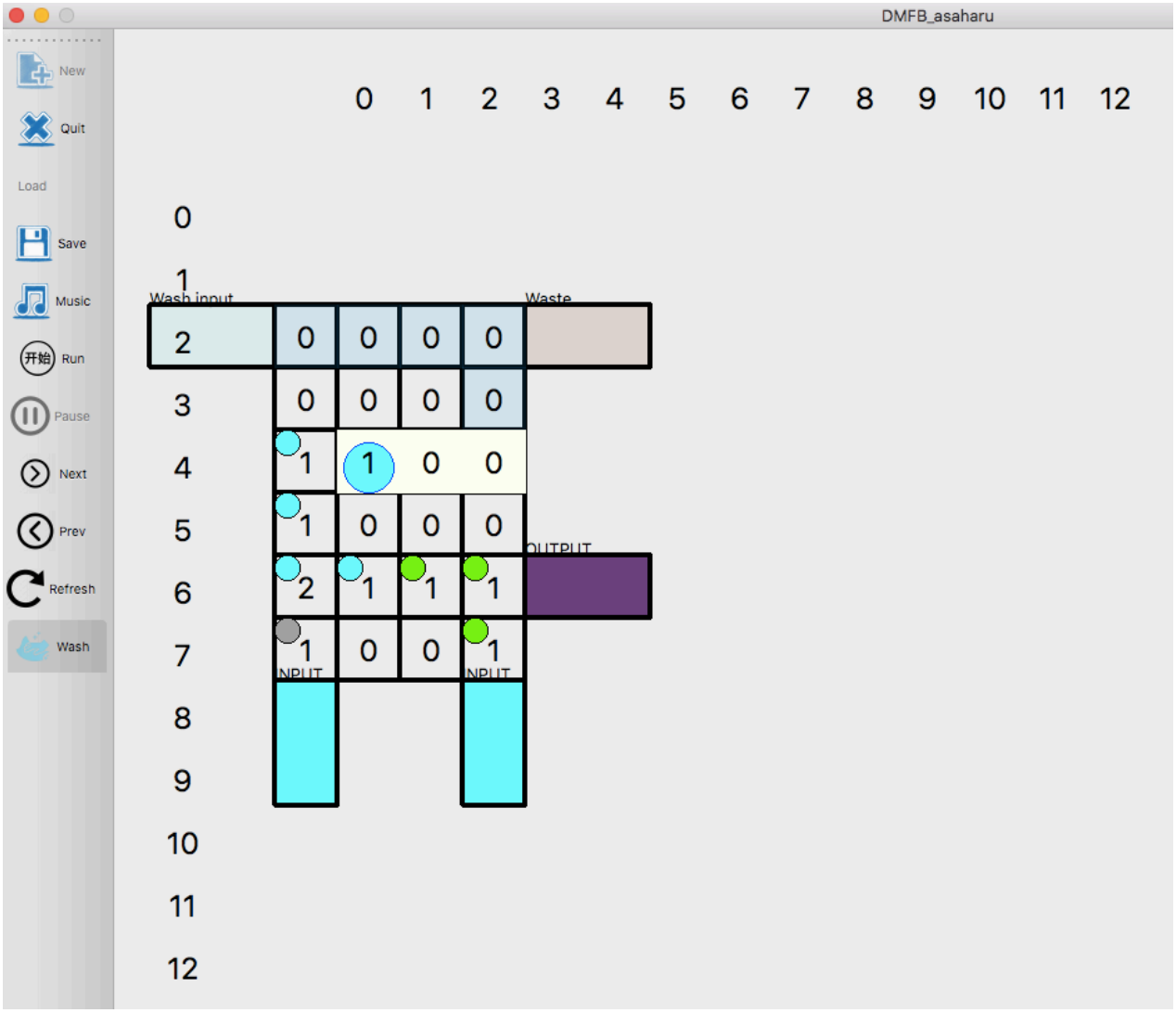

图7 清洗功能示意

`chip.h`中cell类的代码：

```
class Cell {
public:
    Cell();
    ~Cell();

    void clean_waste();
    void set_position(QPoint);
    void drop_reach(Waterdrop*);
    void drop_leave();

    QPoint get_position();
    Waterdrop* get_previous();
    Waterdrop* get_occupied();
    int get_waste_time();
    void change_blocked();
private:
    QSet<int> waster;
    int wastedTime = 0;
```

```
    Waterdrop *occupied;
    Waterdrop *previous;
    QPoint position;
    bool isBlocked = false;
};
```

特色功能

背景音乐

为了增添趣味性，在界面中我增加了music按钮，music按钮能够在点击启动后播放内部src库中的背景音乐，再次点击后能够被关掉。其实现是通过QSound 的play和stop两个API进行实现。

保存界面

此外，有一个save按钮，能够保存当前芯片的时间戳、所有cell的液滴、水滴及污染信息，所有水滴的位置信息以及其他的芯片大小、管道位置信息。再次启动时能够读取保存的这一个时刻的状态。

具体实现是通过本文档开头的hierarchy结构遍历上述实例，将他们的信息写入文件并且输出。

版权说明

本文档说明的工程已经被作者上传至GitHub仓库，如需阅读可以通过用户名的public仓库进行访问。

冯卓尔的GitHub用户名 julius-abu

仓库名称 julius-abu/**dmfb_asaharu**

联系方式： RealAndrewFeng@163.com