数字微流控生物芯片模拟界面设计文档

励永辉

一． 背景简介

数字微流控芯片(DMFB)利用液滴在疏水化表面的电润湿现象，通过控制微电极阵列上的电压变化，操作液滴进行移动(Move)、合并(merge)、分裂(split)和混合(Mix)。数字微流控生物芯片模拟界面可对整个过程进行全程模拟并提供清洗方案。

二． 类的详细介绍

1. Drip 类

```
enum Status{polluted,clean,full,hslipping,vslipping,forbidden};
class Drip
{
public:
    Drip();
    QPoint loc;//坐标
    int color;//颜色
    Status statu;
    QRect getRect();//Drip所在的矩形框坐标
    QPoint getCenter();//中心坐标
};
```

Drip 类用来记录每一个方格的基本信息（颜色、坐标、状态等），其中状态有 polluted（污染）、clean（干净）、full（有水滴）、hslipping（水平分裂）、vslipping（竖直分裂）、forbidden（禁止放置）。

2. Info 类

```
class Info
{
public:
    Drip drips[110][16][16];

    int row,column;
    int endtime;
    int time;
    int warterTot; //water总数
    QSet<int> pollutecolor[16][16];
    QVector<QPoint> washstep;
    QVector<QPoint> Input;
    QVector<QPoint> Output;
    QSet<QString> Musicrecord[110];
    bool isautoplay;
    bool washable[16][16];
    bool haswarning;
    bool isconnect;
    bool iswash;
    int washtime;
    int dangeroustime;


    void changetime(bool isadd);
    bool BFS(QPoint startP,QPoint endP,QVector<QPoint>& shortestPath);
    bool getLoc(int &a,int &b,QPoint pos);
    void playmusiccontrol(int t);
    void init();
    void dangerdetect();
    void readtxt(QString filename);
    Info();
};
```

Info 类是储存和处理信息的枢纽。Drips[i][j][k]储存着 i 时刻坐标为（j，k）的格子信息。QSet<QString> pollutecolor 储存污染过每个格子的所有颜色。QVector<Qpoint> Input 储存输入端口，QVector<Qpoint> Output 储存输出端口。QVector<QPoint> washstep 储存清洗时的路径，bool washable[i][j]储存（i，j）处是否被用户设置了清洗障碍。

3. Newpaint 类

```
class NewPaint
{
public:
    NewPaint(Info* info);
    void draw();//在device界面上按照info的信息绘制游戏界面
    void drawPut();
    void drawNum(); //结束后绘制污染次数
    void drawWashPut();
    void drawBarrier();
    void drawWash();
    void drawDrip(QColor color,QPoint pos,Status statu);
    void drawSlippingDrip(QColor color,QPoint pos,bool ishorizon);
    QPainter *painter;


private:
    void drawGridding();//画网格线
    void drawDrips();//画水滴
    QPaintDevice *device;
    Info *info;
};
```

NewPaint 负责将 Info 中储存的信息在 MainWindow 中绘画出来。其中 draw 为主要控制函数，来调用其他绘制函数。drawGridding、drawDrips、drawBarrier 等函数分别用来绘制特定部件。

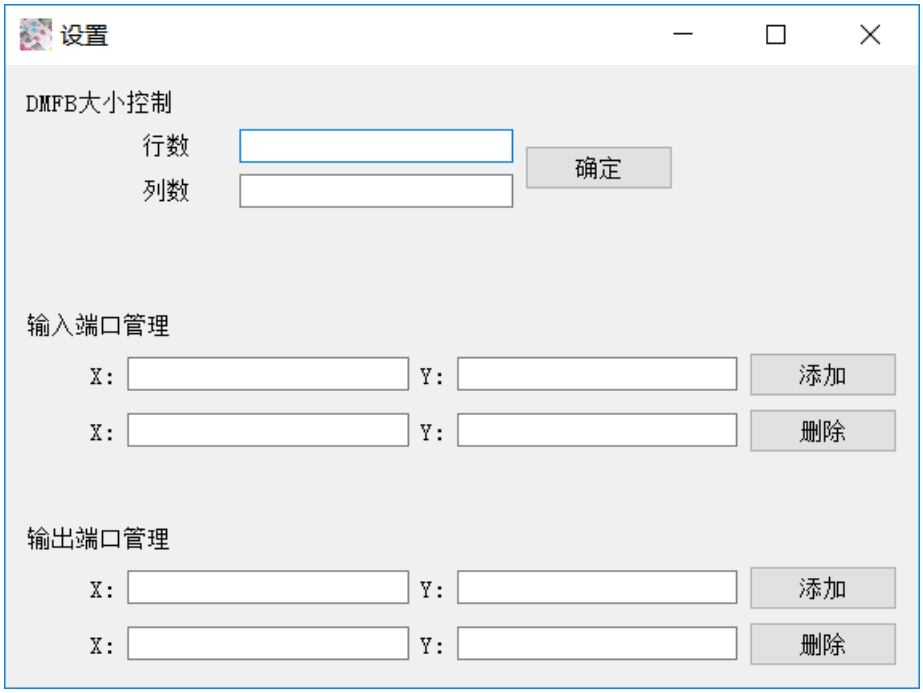
4. Music 类

```
class Music
{
public:
    Music();
    void loadMusic();//载入音乐
    void playMusic(QString name);//播放音乐name

private:
    map<QString,QString> musicList;
    QSound *background;
};
```

Music 类负责播放音效。loadMusic 函数负责提取音频文件的路径到 map<QString,QString> musicList 中。playMusic 函数负责播放音乐。

5. setting_interface 类

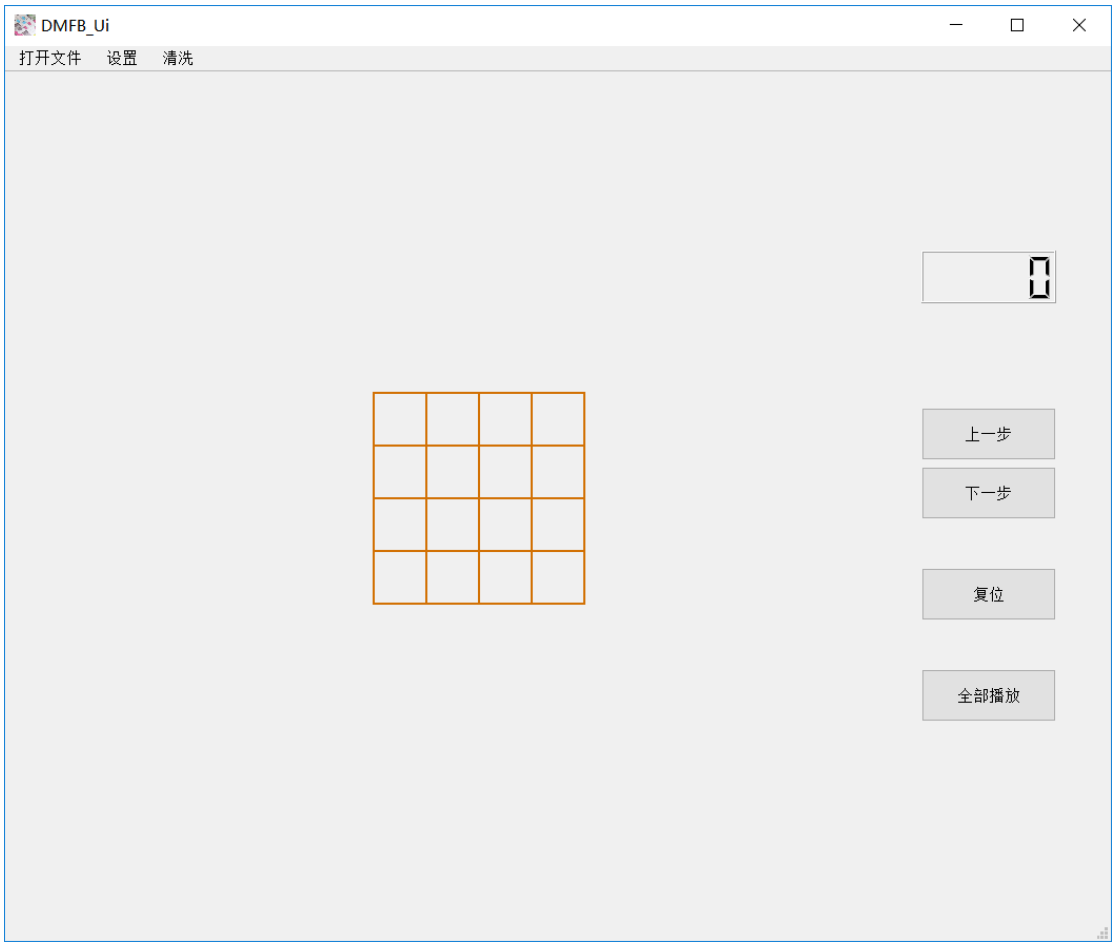
```
namespace Ui {
class setting_interface;
}

class setting_interface : public QWidget
{
    Q_OBJECT

public:
    explicit setting_interface(QWidget *parent = 0);
    ~setting_interface();
signals:
    void sentDMFBchange(int now_row,int now_column);

    void sentInputChange(int row,int column,bool isadd);

    void sentOutputChange(int row,int column,bool isadd);

private slots:
    void on_DMFBButton_clicked();

    void on_InputAddButton_clicked();

    void on_InputDeleteButton_clicked();

    void on_OutputAddButton_clicked();

    void on_OutputDeleteButton_clicked();



private:
    Ui::setting_interface *ui;
};
```



setting_interface 类用来设计设置界面，主要包含几个 signal 和 slot，用来捕获用户输入的信息，并把它传递出去。

6. MainWindow 类

```
namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
    void paintEvent(QPaintEvent *event);
    void mousePressEvent(QMouseEvent *event);
    void wash();


private slots:
    void on_setting_action_triggered();

    void on_fileopen_action_triggered();

    void DMFBchange(int now_row,int now_column);

    void InputChange(int row,int column,bool isadd);

    void OutputChange(int row,int column,bool isadd);

    void on_BeforeButton_clicked();

    void on_AfterButton_clicked();

    void on_RestartButton_clicked();

    void on_AutoButton_clicked();

    void onTimeOut();

    void onWashTimeOut();

    void on_wash_action_toggled(bool arg1);

private:
    Ui::MainWindow *ui;
    Info info;
    NewPaint *newpainter;
    QTimer *timer;
    QTimer *timer2;
};
```



MainWindow 类用来设计主界面，由菜单栏、绘画区域、lcd 显示器和按钮构成。包含了几个 slot 来处理用户的操作。并且重载了 paintEvent, 使用自定义的 Newpaint 绘制界面；重载了 mousePressEvent 来捕获用户鼠标左键点击来设置/取消清洗障碍。

三． 设计思路

整个程序包含个板块，分别为信息储存和处理模块（Defaultparameters，Drip，Info）、界面模块（MainWiondow，setting_interface）、绘图模块(NewPaint)和音乐模块(Music)。

以下将依据功能解释设计思路。

读取数据时，把每个时刻每个格子的信息都储存在 Drips[i][j][k]中。文件中的操作即改变某些时刻的各自状态。例如 Move t,x1,y1,x2,y2 即为将 t 时刻后(x1,y1)位置的格子状态都设定为 pollute，将 t 时刻后(x2,y2)位置的格子状态都设定为 full。然后播放时改变时间 time 来分别读取不同时刻的全部格子状态。并把每个时刻需要播放的音乐记录在 QSet<QString> Musicrecord; 把每个格子的所拥有过的所有颜色信息都记录进 QSet<int> pollutecolo[ ][ ]中.

“下一步”、“上一步“、”全部播放“、复位等按钮即改变 time 的状态从而在主界面显示不同时刻的格子状态。

播放音乐时从 Musicrecord 中读取每个时刻需要播放的音乐，并用 playmusic（）函数来调用音频文件。

移动约束检查每个液滴 t 和 t+1 时刻所处的九宫格中是否存在其他液滴。

QSet<int> pollutecolo[i][j]中记录了（i，j）格子所拥有过的全部颜色信息，其 size 即

为该格子的污染次数，在播放完成后显示在该格子上。

打开清洗功能后，每个时刻都先检测被 pollute 的格子并记录 QVector<QPoint> needtowash 到，然后用 BFS（）搜索 needtowash 中格子的相互路径。然后利用静止时间功能，清洁水滴从 Wash Input 进入，通过搜索的路径行走，最终从 waste 端口离开。

四． 遇到的问题以及解决方案

1. BFS()搜索过程耗时相对较长，而某些函数是与 QTimer timer 相连定时运行，如果这些函数调用了 BFS()的输入结果，就会导致程序崩溃。
   解决方案：在 BFS()运行时用 QThread::msleep()函数阻碍其他某些函数的运行过程。

2. 调整窗口布局时候需要修改很多地方的坐标，很可能会乱套。
   解决方案：新建 defaultparameters.h 来记录全局变量，方便修改。

3. 页面中的绘画有时候会出现漂移
   解决方案：在每个坐标转移后面都恢复到原坐标系，避免 update（）等函数多次调用导致坐标多次转移。