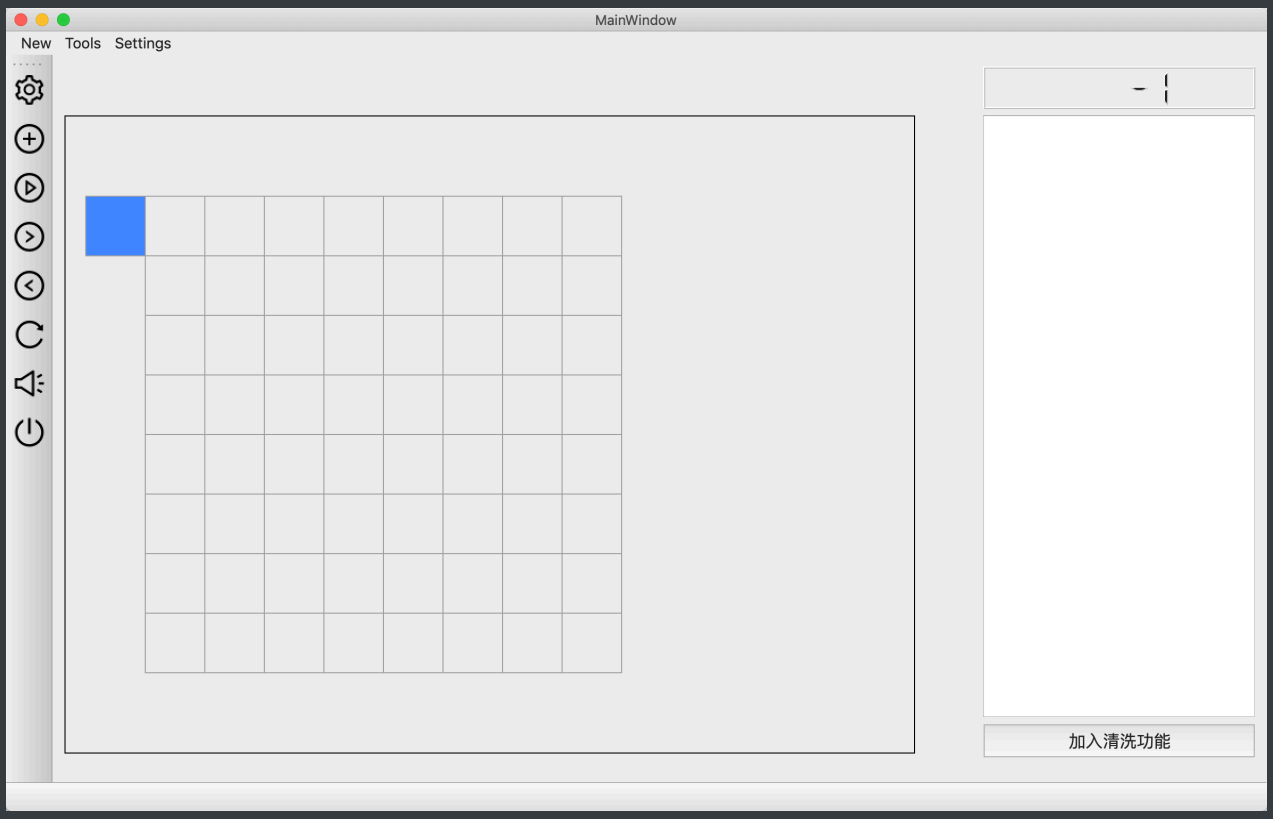
说明文档

1.使用方法

程序进入界面



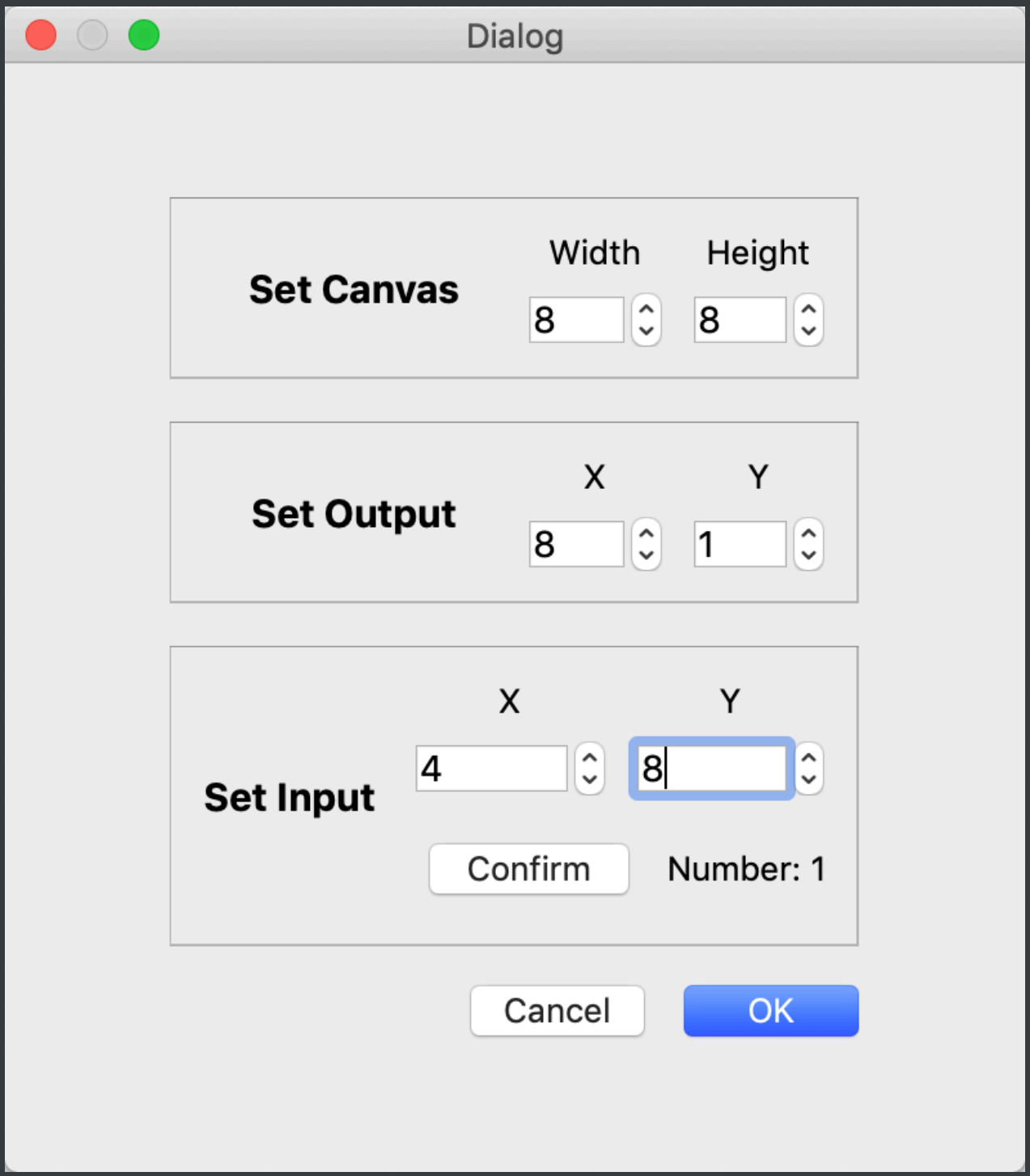
可以看到左侧的工具栏，上部的菜单栏，左侧的网格线，右侧依次排列的是计时器，命令显示窗口，清洗功能选择按钮。

左侧工具栏从上之下依次为：

1. 设置网格的基本宽高及Input Output位置
2. 读入命令文档
3. 开始连续播放画面
4. 单步向后执行命令，每次点击时间加一
5. 向前一步
6. 重置时间及网格状态
7. 选择声音是否播放
8. 退出程序

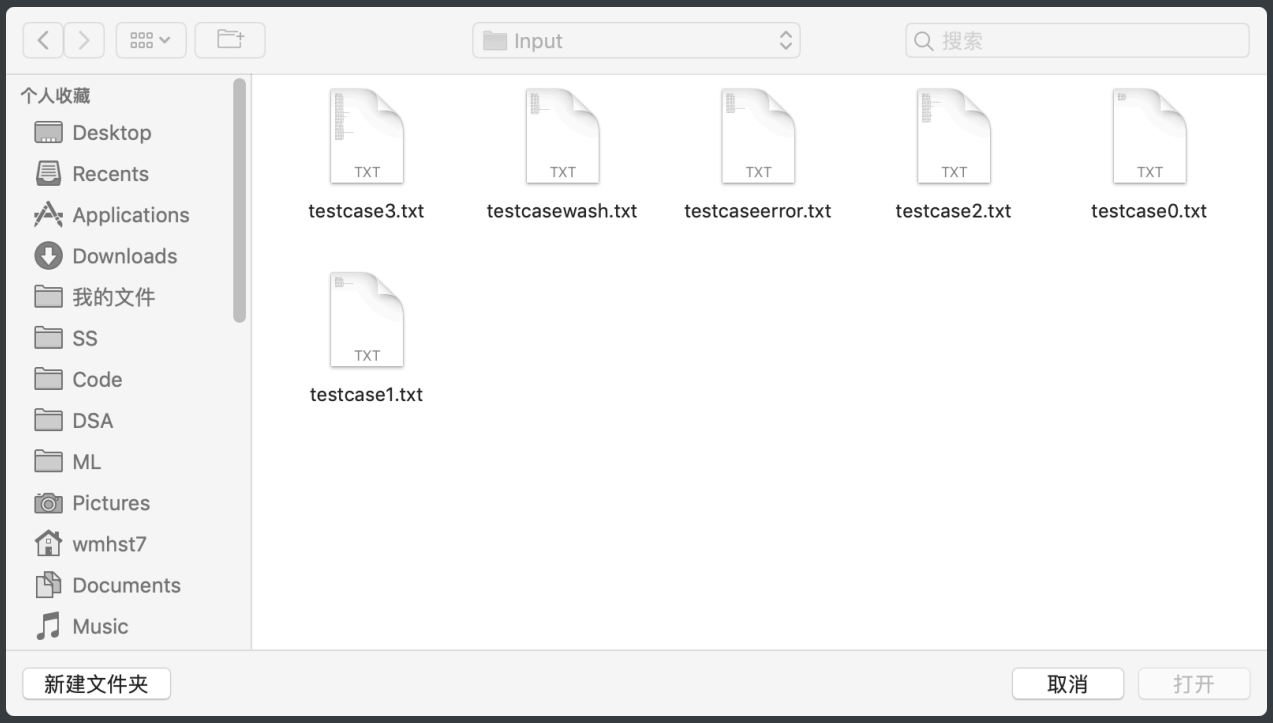
以下简述程序正常运行的步骤

首先设置网格基本属性

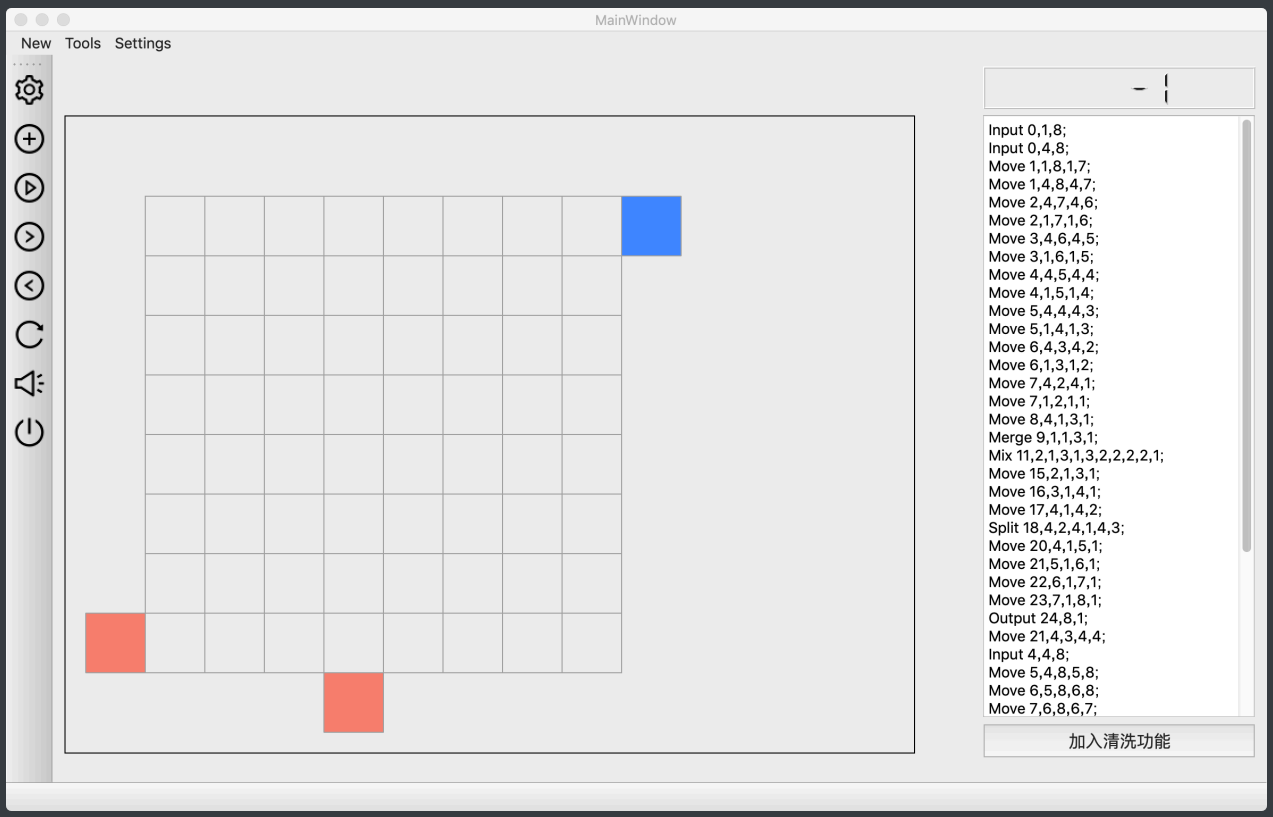


注意设置Input时，每次添加一个Input位置需点击Confirm键来保存，设置完毕后点击OK确认。

读入命令文件



弹出文件选择窗口，选择文件即可。

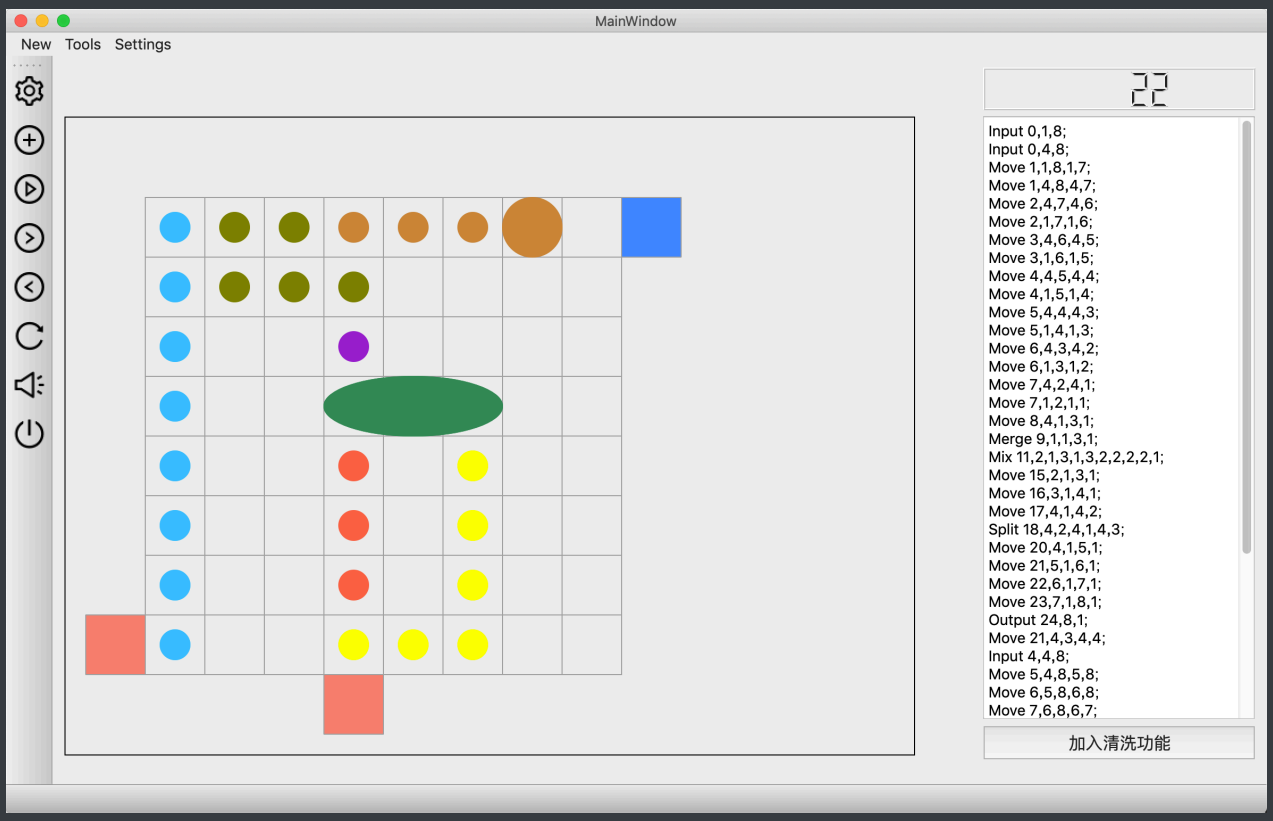


选择后右侧显示命令。

普通执行

点击  单步执行或者点击  连续执行。

执行中画面

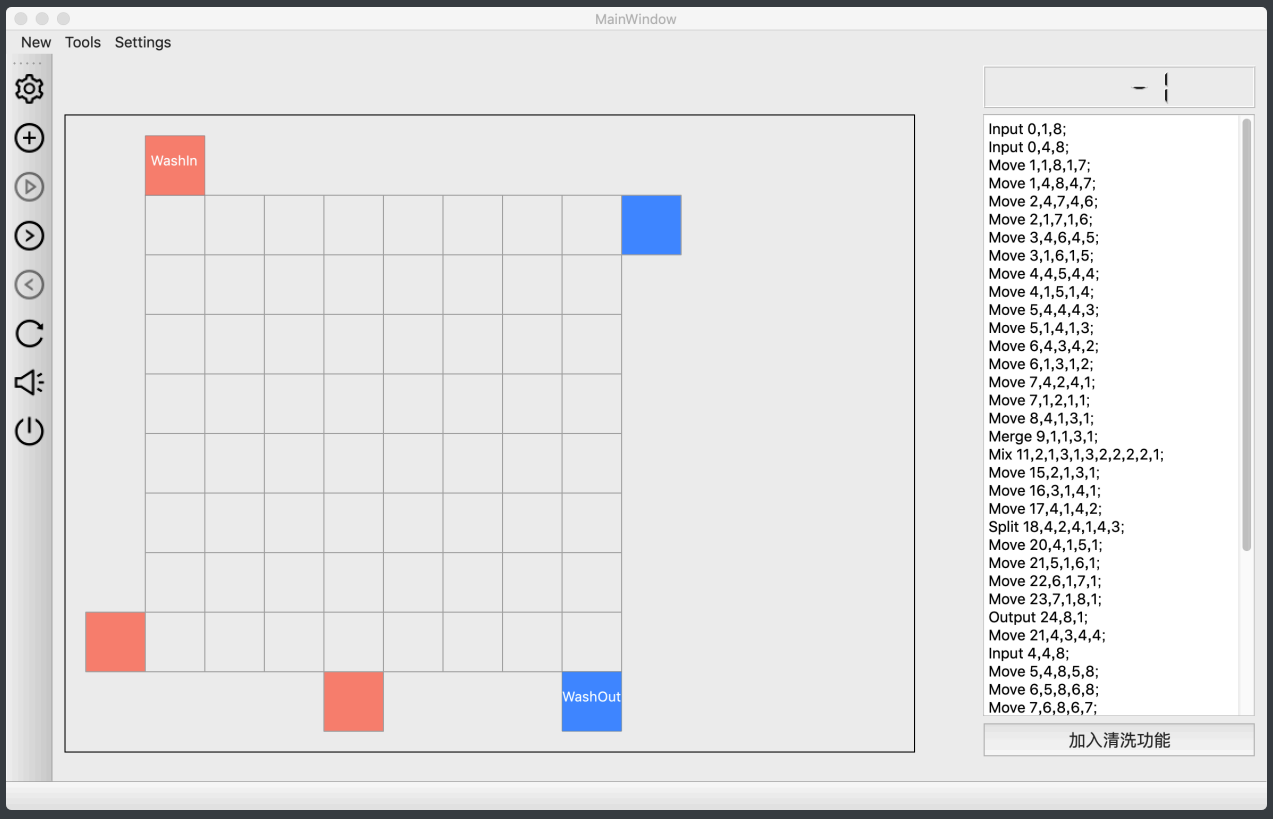


此时可以选择是否开启声音以及是否重置，点击上一步可回退执行。

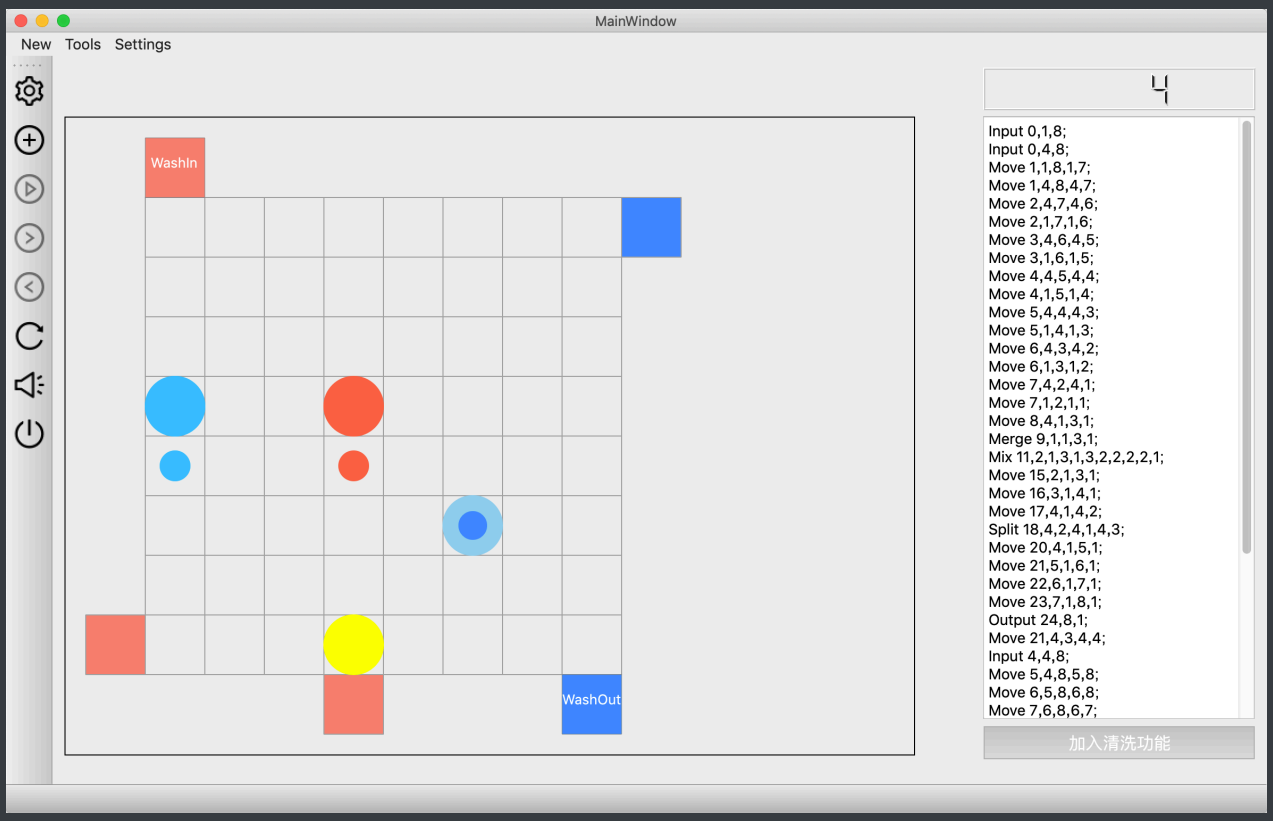
当程序执行完所有命令后显示污染次数。

带清洗功能的执行

选择"加入清洗功能"，显示清洗Input与Output

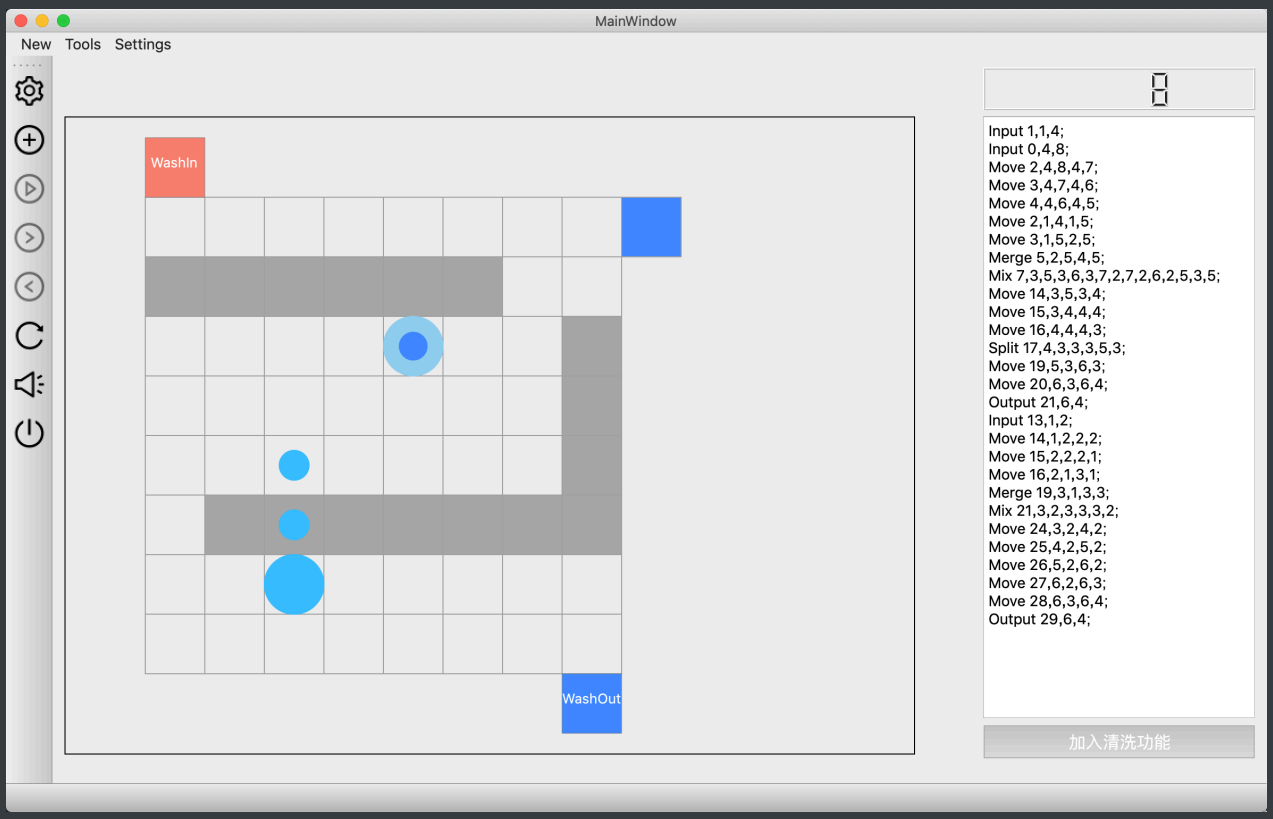


此时只有"下一步"可用，点击可出现自动移动的清洗液滴（同心圆）。



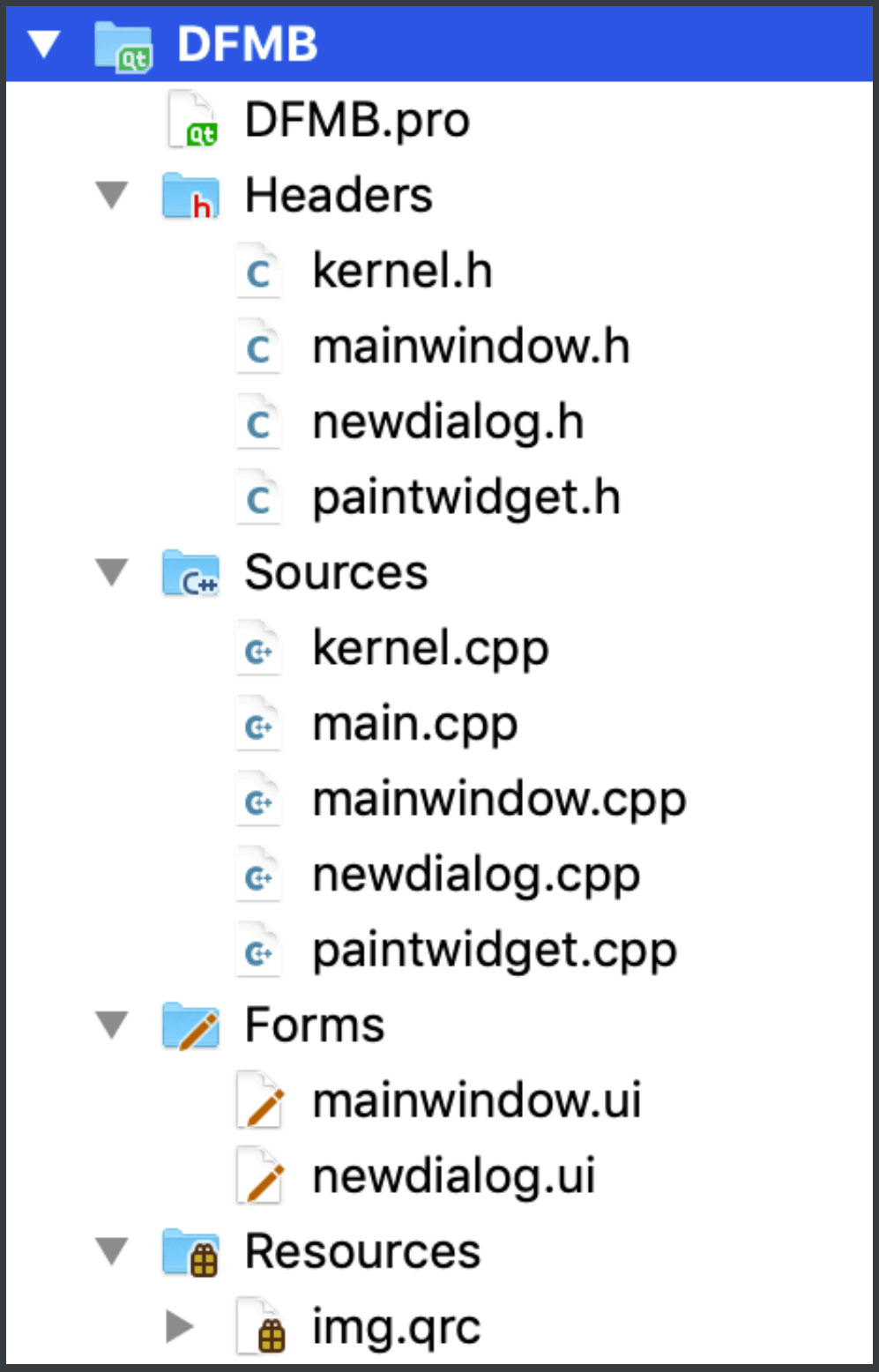
清洗功能可以取消。

清洗过程中可以点击方块添加障碍，在下一次执行"下一步时"会自动避开此障碍。



2.设计说明

程序文件树：



其中主要有以下类：

```
class Kernel : public QObject//管理数据
{
    Q_OBJECT
public:
    Cube CubeData[14][14];//格子的数据
    int Width = 8;
    int Height = 8;
```

```
    QVector<Position> InputPositions;
    Position OutputPosition;
    QStringList Orders;
    int Time = -1;//时间
    int Interval = 700; //间隔时间ms
    int KindNumber = 0;//液滴总数
    bool Sound = true;
    bool Wash = false;//是否清洗状态
    QVector<QString> ColorName;
};

class Cube{//单个格子的数据
public:
    Position Other = Position(0, 0);//用于Split与Merge中间状态
    int kind = -1;//now drop kind
    bool Expanding = false;//now ecllipse
    bool blocked = false;//是否是阻挡方块
    bool Washing = false;//是否是清洁液滴
    QVector<int> PolluteKinds;//Polluted Drop Kinds
};

struct Position{//封装位置数据
public:
    int X;
    int Y;
};

class PaintWidget : public QWidget//继承自QWidget的自定义类，加入数据管理与绘图功能
{
    Q_OBJECT
public:
    explicit PaintWidget(QWidget *parent = nullptr);
    void paintEvent(QPaintEvent *);//绘图函数

    Kernel * kernel = nullptr;
    QVector<Position> InputPositions_draw;
    Position OutputPosition_draw = Position(0,1);
    bool end = false;//是否结束

};
```

程序有两个UI窗口：一个主界面，一个用来设定基本属性的对话框。

数据管理与操作在Kernel类中实现，UI界面类（MainWindow类，NewDialog类）中有指向Kernel的指针，用于实现UI界面与数据内核的分离。

操作事件一般会触发UI界面类的槽函数，进而调用kernel里对应的操作槽函数。