高级语言程序设计

实验报告

南开大学 计算机学院

姓名 林语盈

学号 2012174

班级 2020 级计算机卓越班

2021 年 5 月 8 日

目录

高级语言程序设计大作业实验报告

一. 作业题目

电路图绘制软件。

代码仓库链接：林语盈/CircuitDiagram (gitee.com)

二. 开发软件

QT 6.0.1

QT creator 4.14.0

编程语言：C++

三. 课题要求

1) 面向对象。

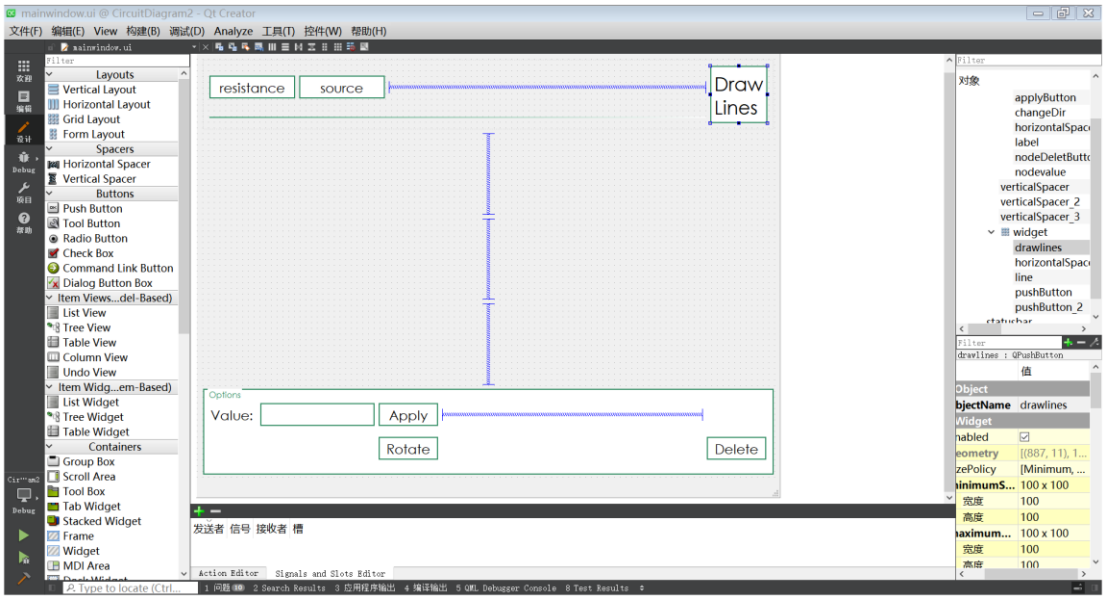
2) 单元测试。

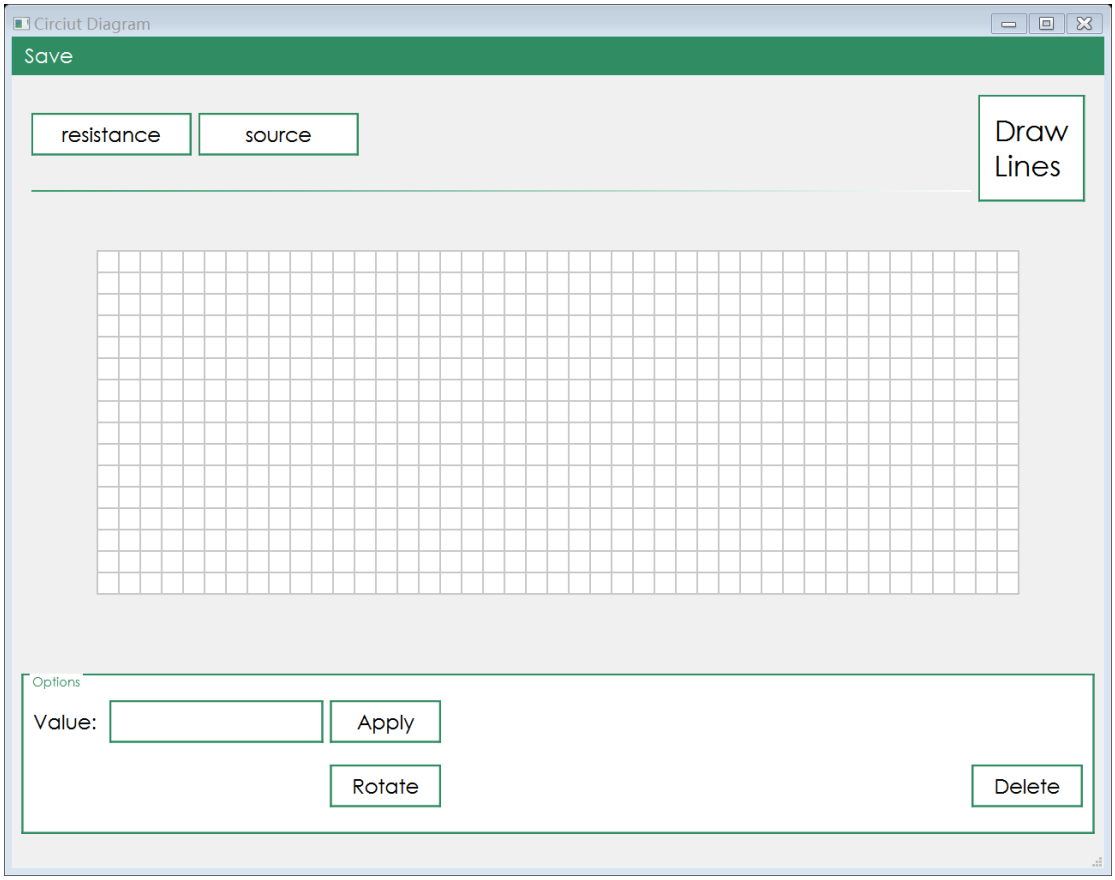
3) 模型部分

4) 验证

四. 主要流程

1．UI 设计



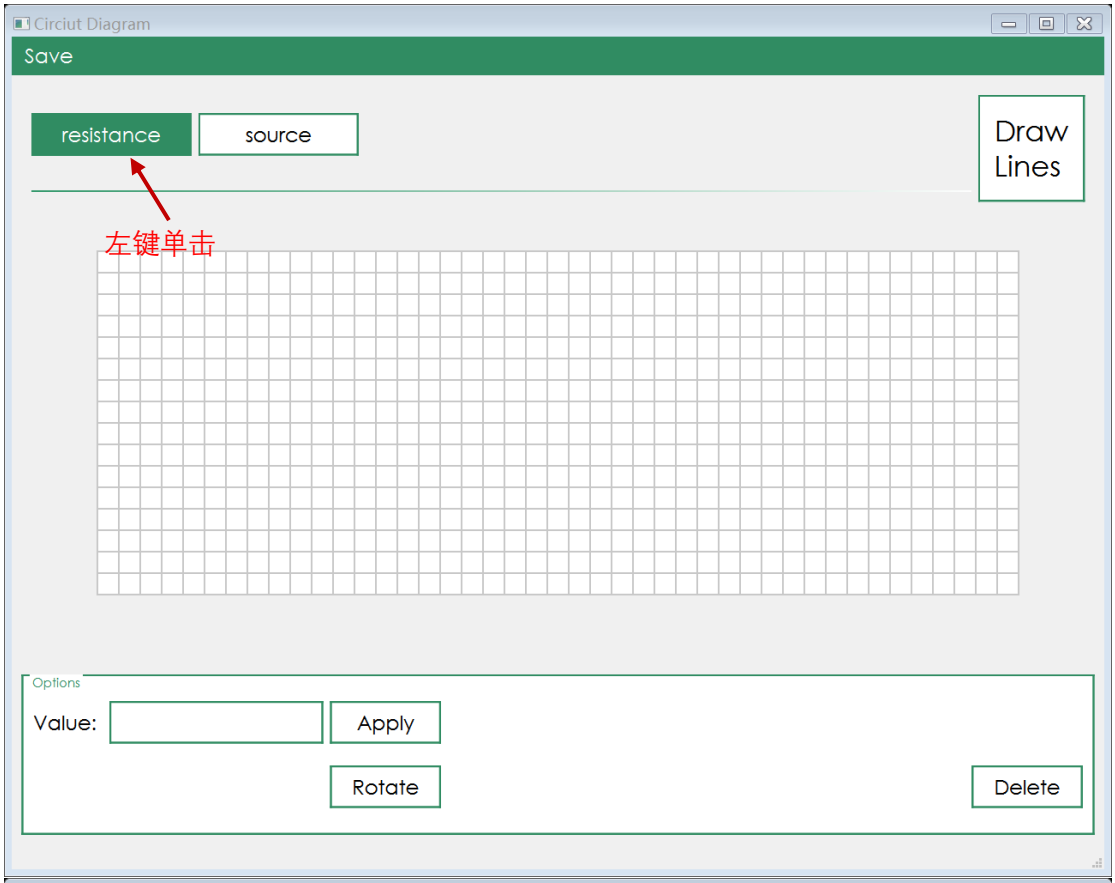


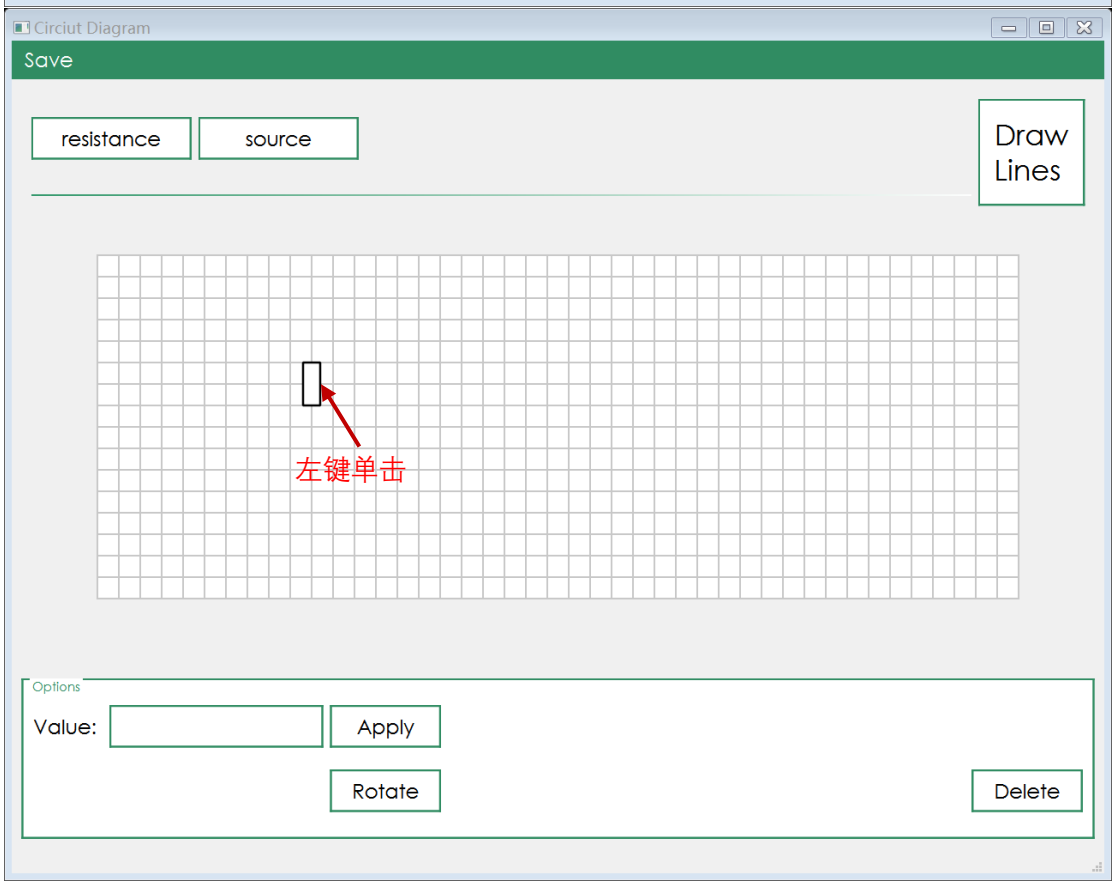
2．用例设计

（1）添加元件

① 左键单击相应元件按钮，按钮进入已点击状态。

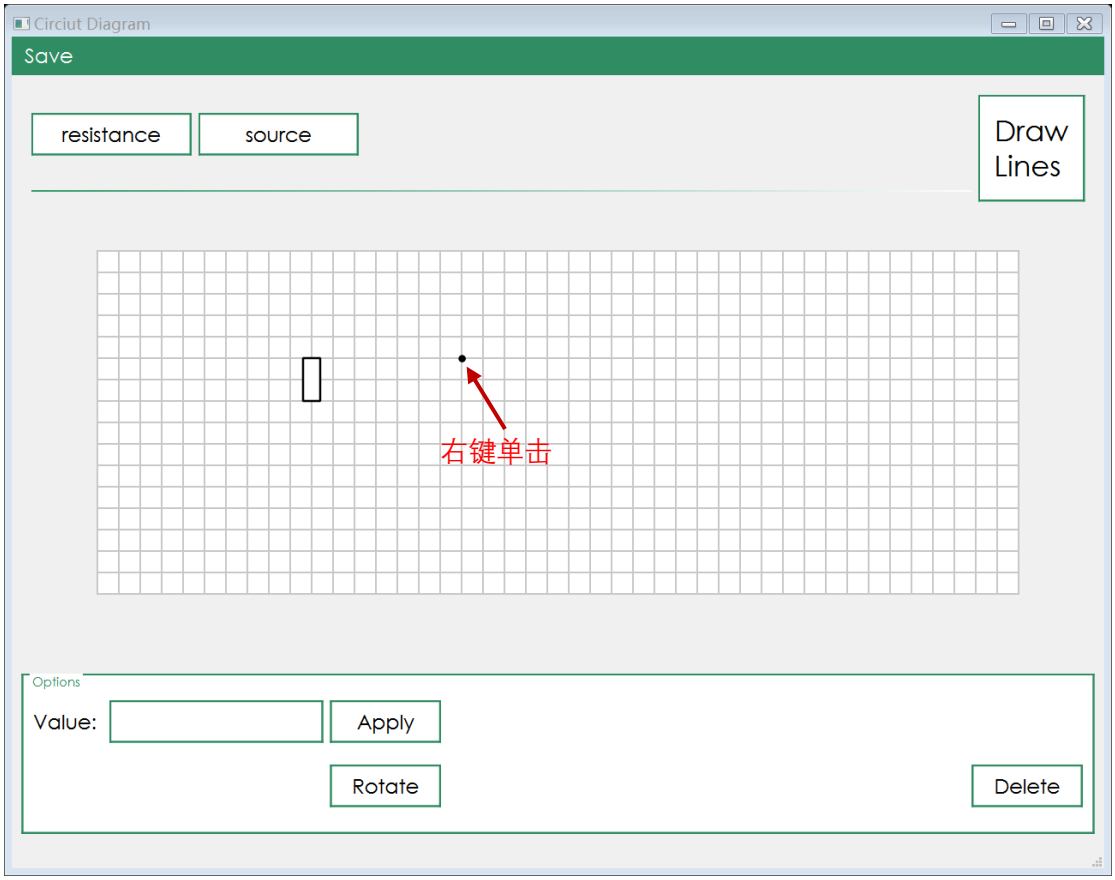
② 点击元件位置，出现原件


Circiut Diagram
Save
resistance
source
Draw Lines
左键单击
Options
Value:
Apply
Rotate
Delete


Circiut Diagram
Save
resistance
source
Draw Lines
左键单击
Options
Value:
Apply
Rotate
Delete

(2)添加结点

①在画板区域，单击右键，添加节点。



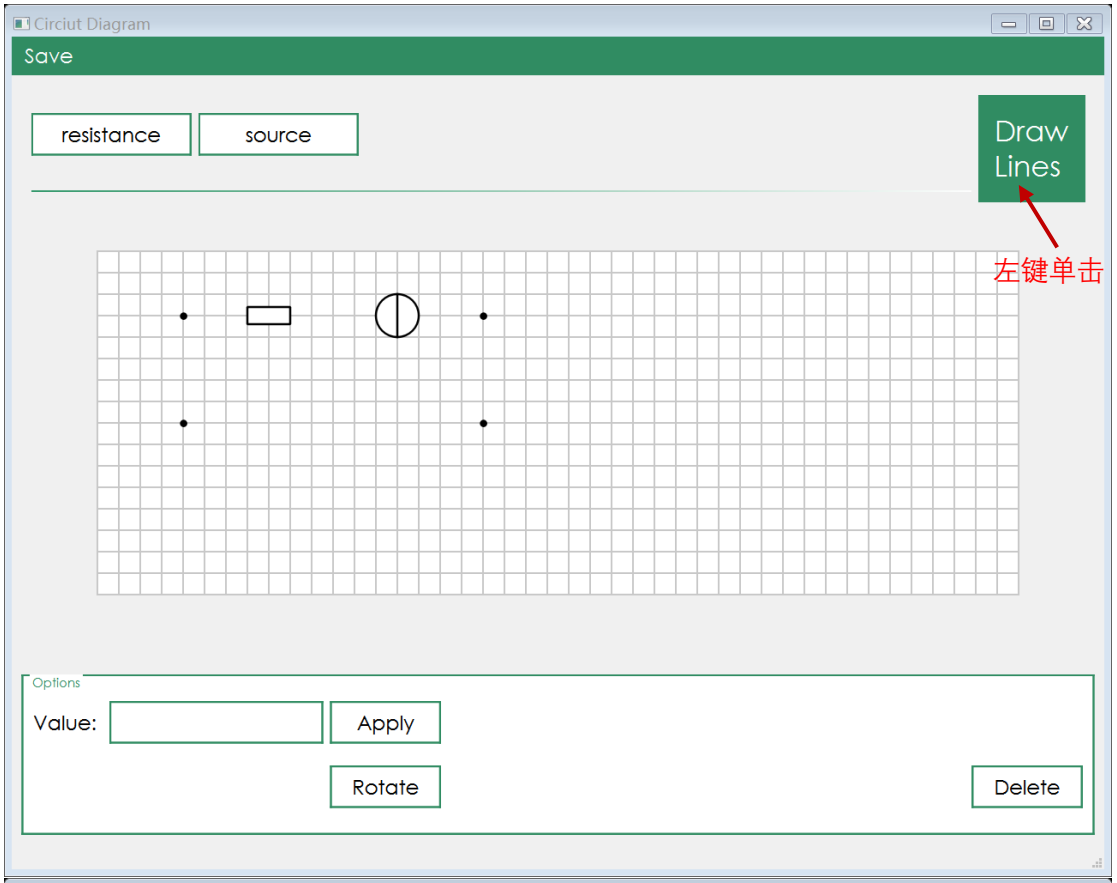
（3）添加连线

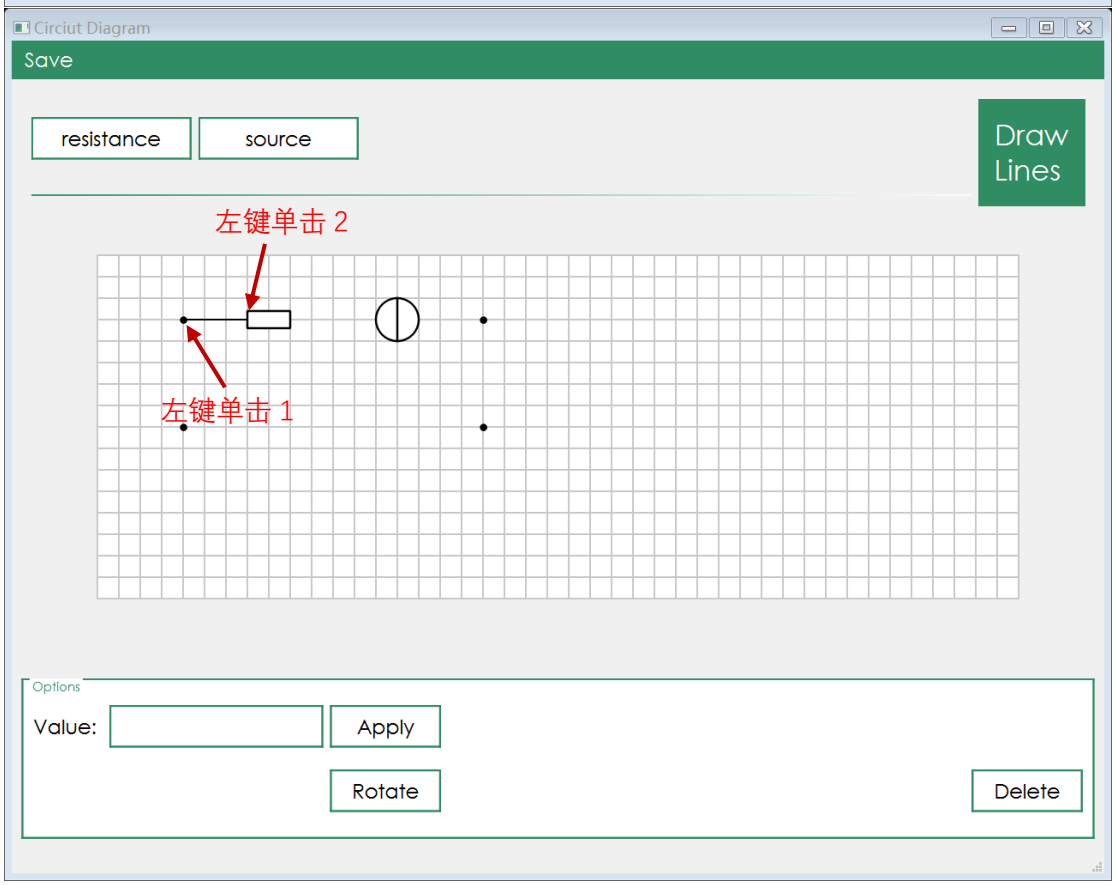
①左键单击“draw lines”按钮，进入连线模式

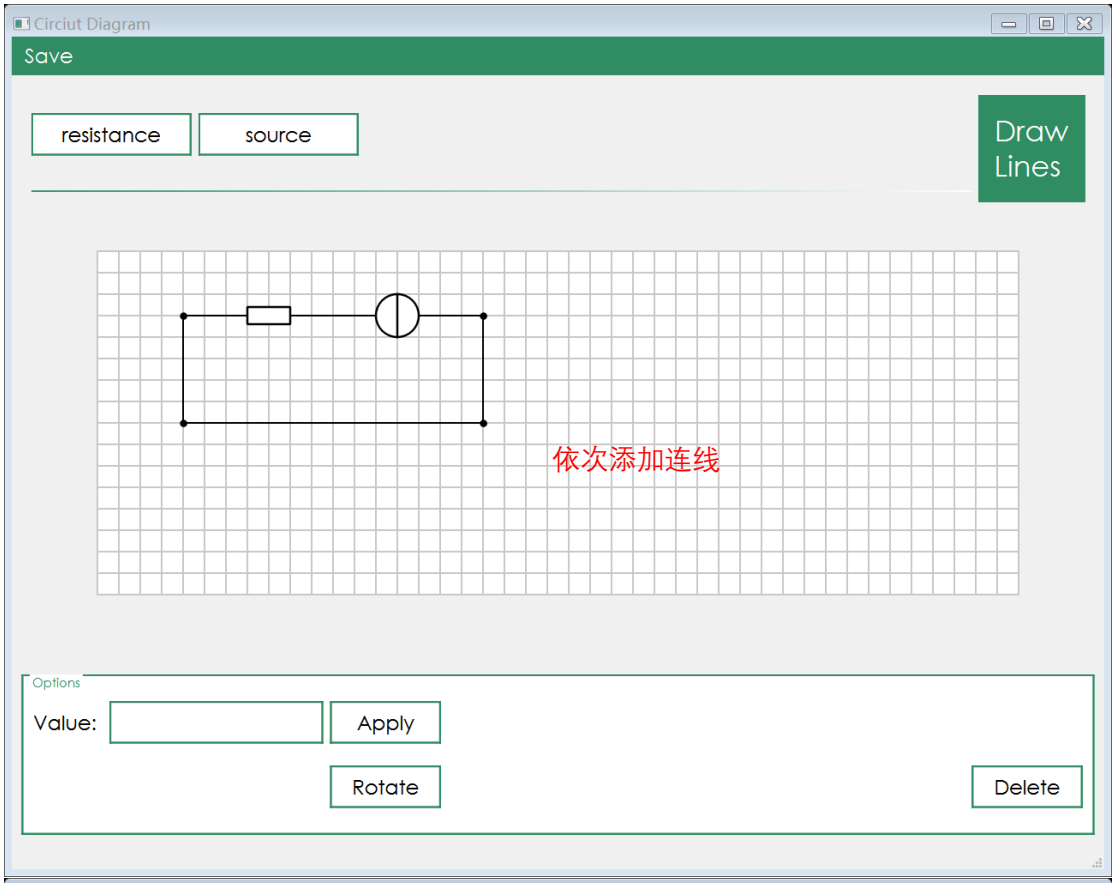
②依次点击结点或元件四周（连线点），连线

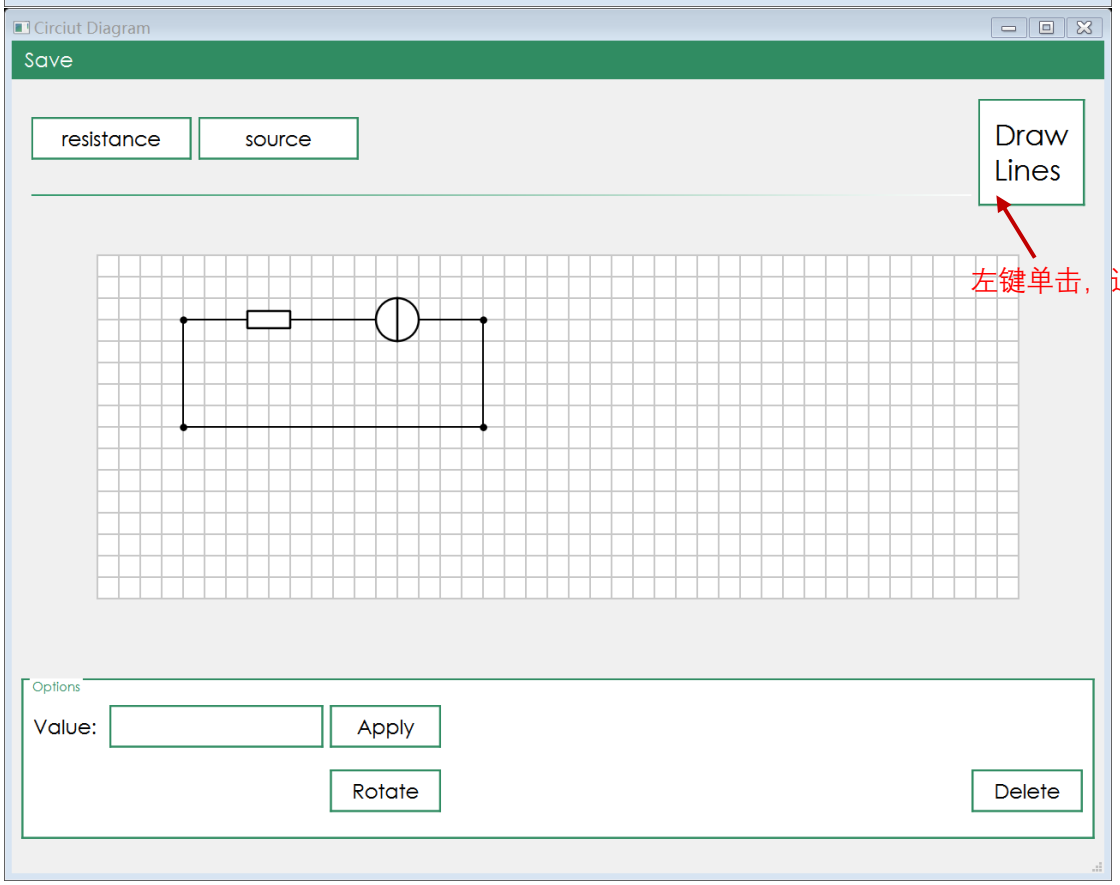
③重复上一步操作

④左键单击“draw lines”按钮，退出连线模式


Circiut Diagram
Save
resistance
source
Draw Lines
左键单击
Options
Value:
Apply
Rotate
Delete

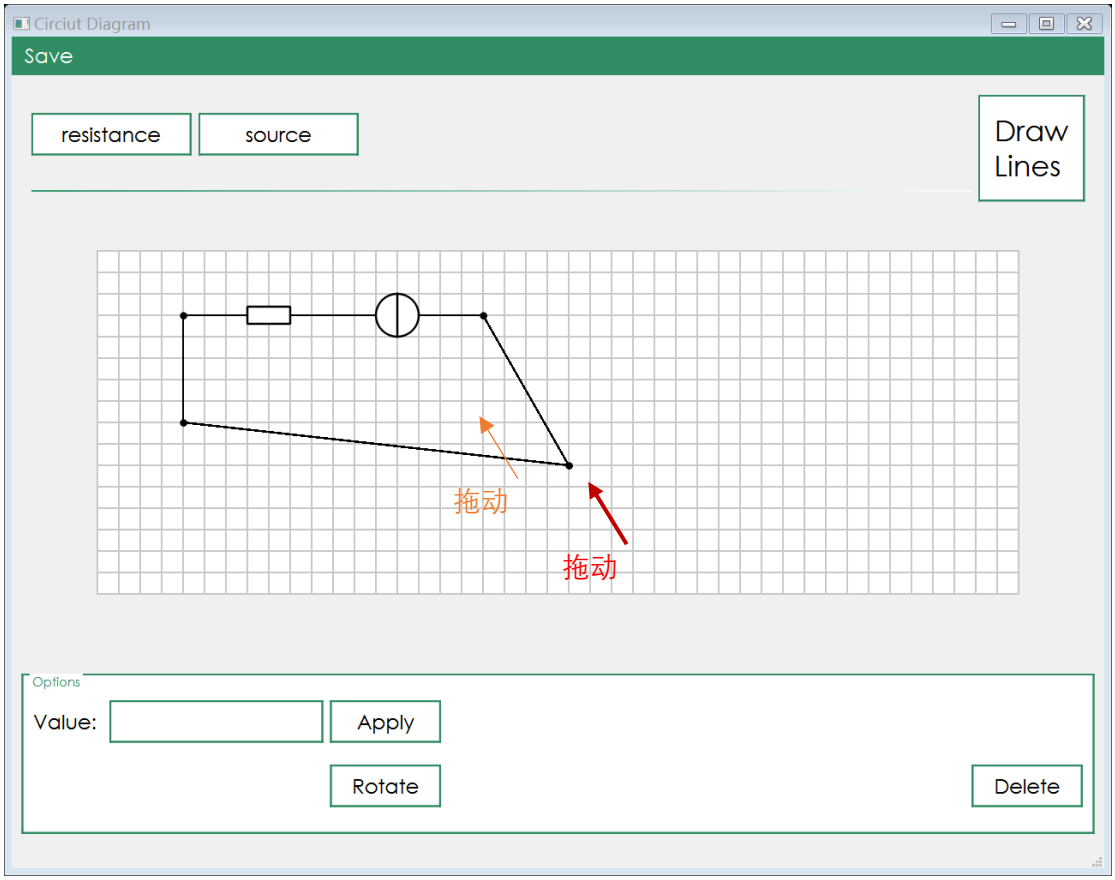

Circiut Diagram
Save
resistance
source
Draw Lines
左键单击 2
左键单击 1
Options
Value:
Apply
Rotate
Delete


Circiut Diagram
Save
resistance
source
Draw Lines
依次添加连线
Options
Value:
Apply
Rotate
Delete


Circiut Diagram
Save
resistance
source
Draw Lines
左键单击，退出连线模式
Options
Value:
Apply
Rotate
Delete

（4）拖动元件

①使用鼠标左键拖动元件，元件高亮，连线将自动跟随。

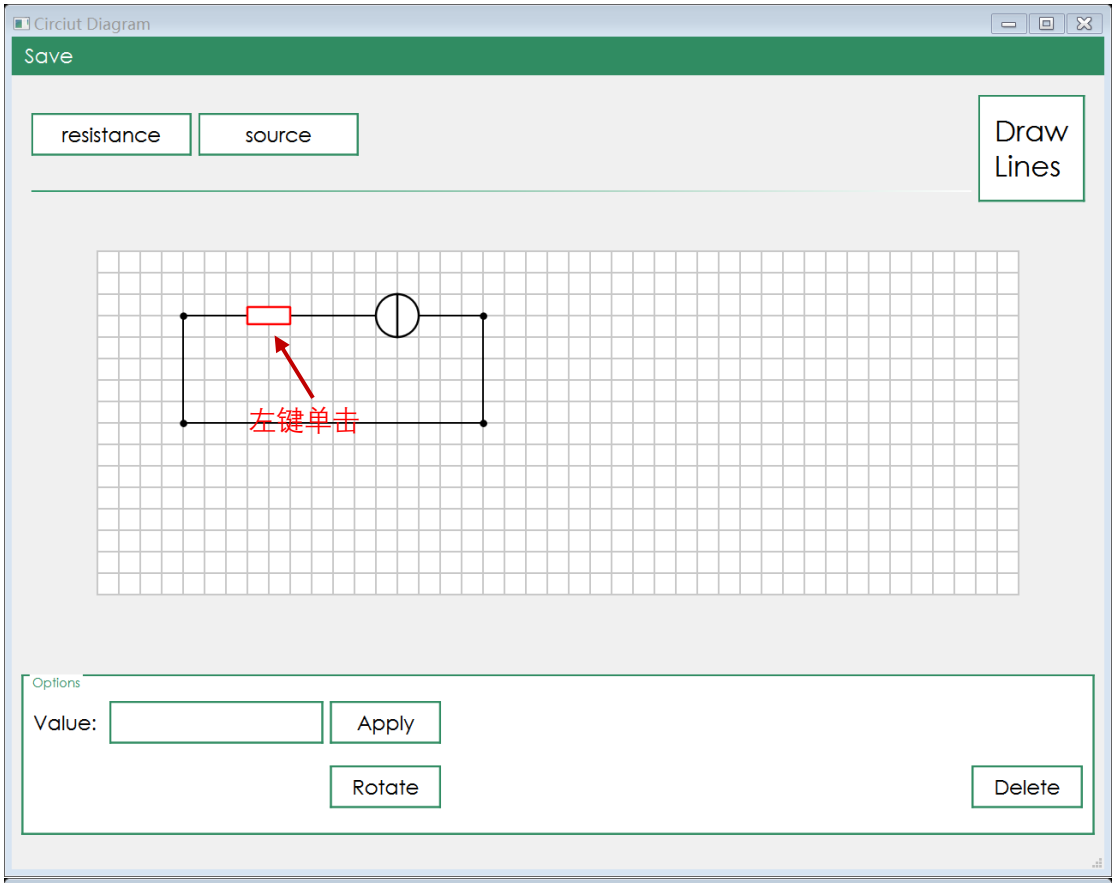


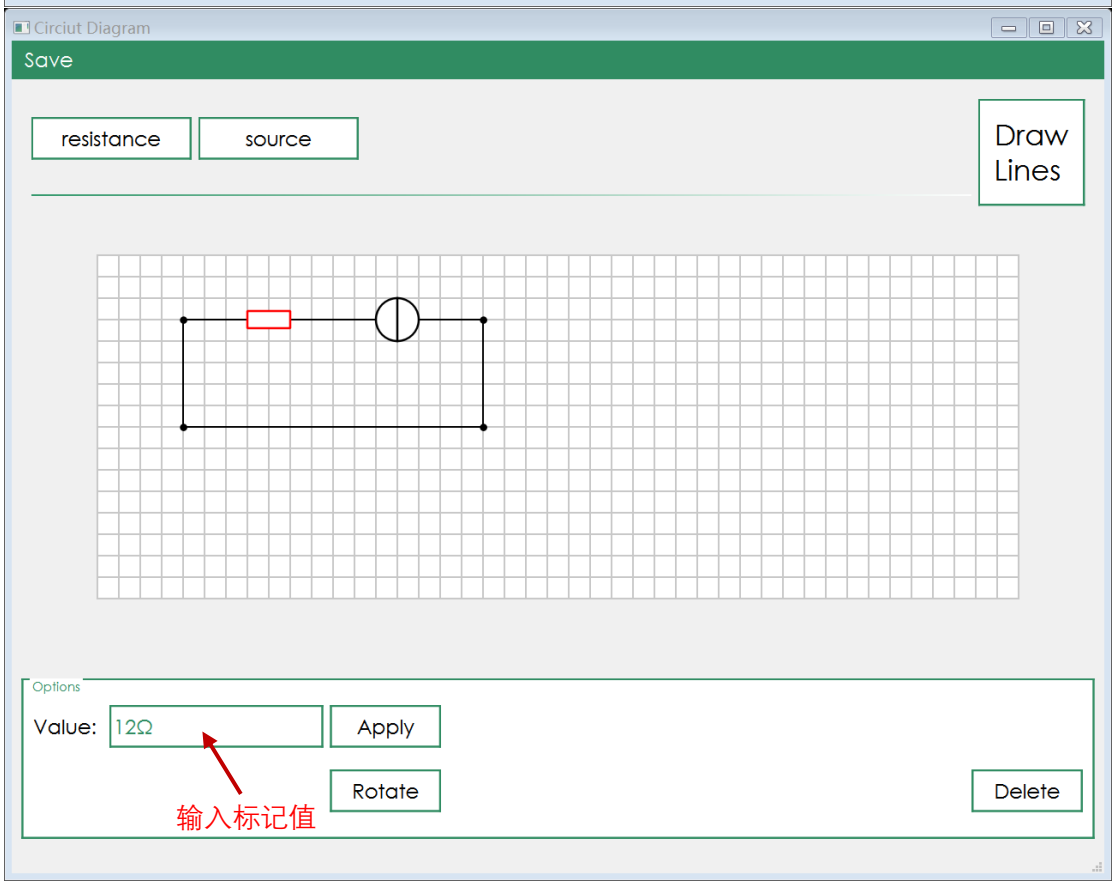
（5）更改元件标记值

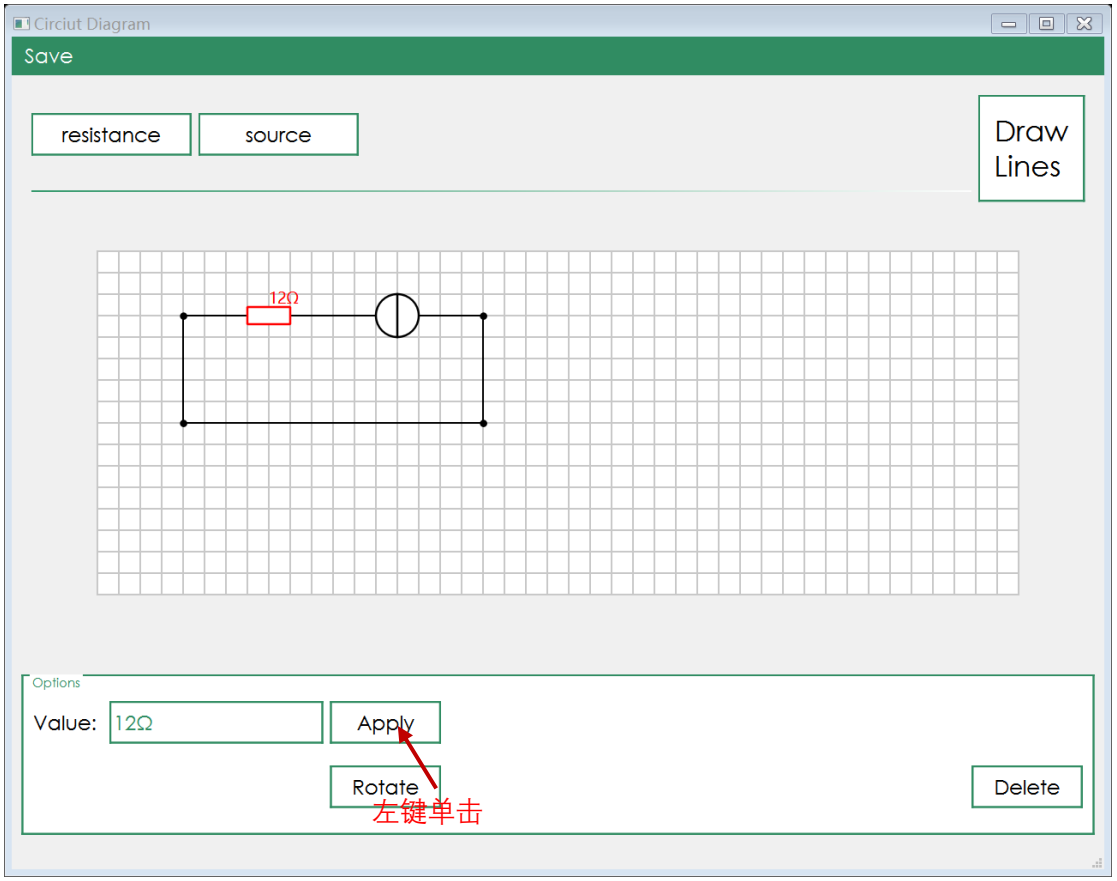
①左键单击选择元件，元件高亮显示

②在“Value”框中输入元件标记值（电阻阻值等）

③左键单击“Apply”按钮，改变原件标记值。


Circiut Diagram
Save
resistance
source
Draw Lines
左键单击
Options
Value:
Apply
Rotate
Delete

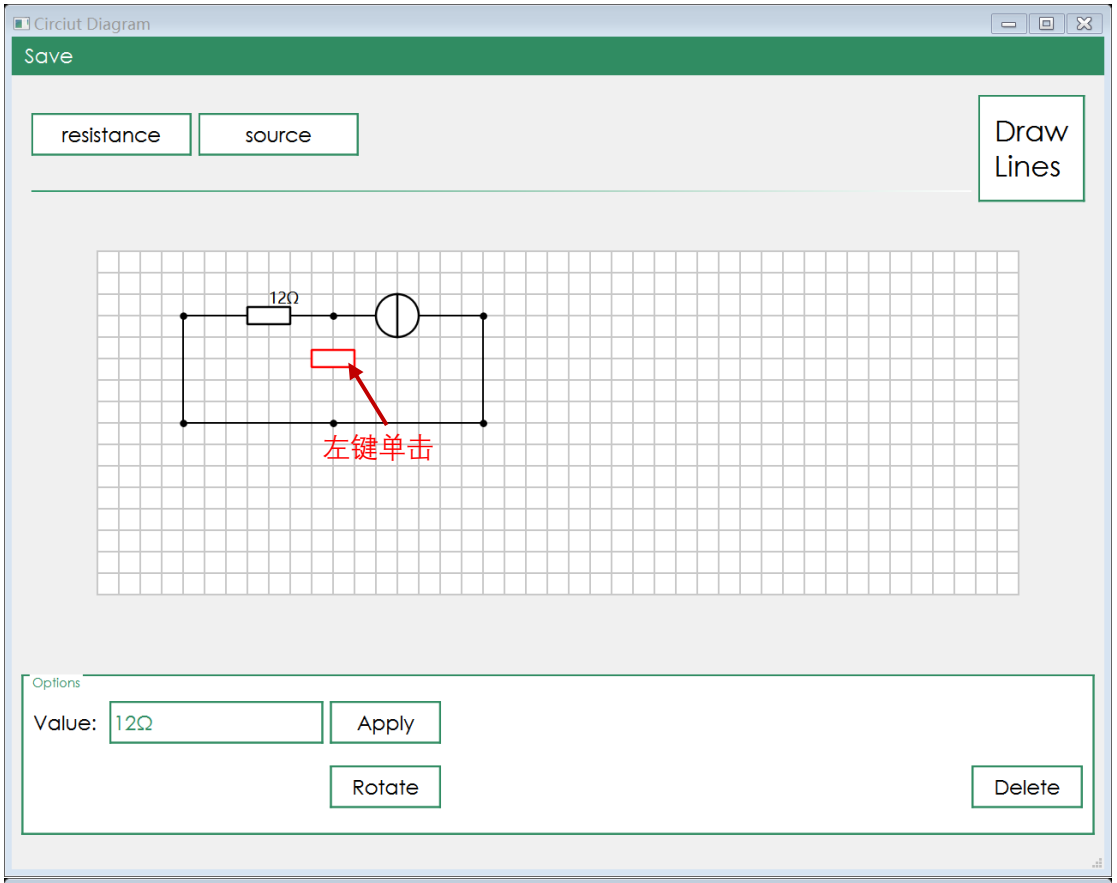

Circiut Diagram
Save
resistance
source
Draw Lines
Options
Value:
12Ω
Apply
Rotate
Delete
输入标记值

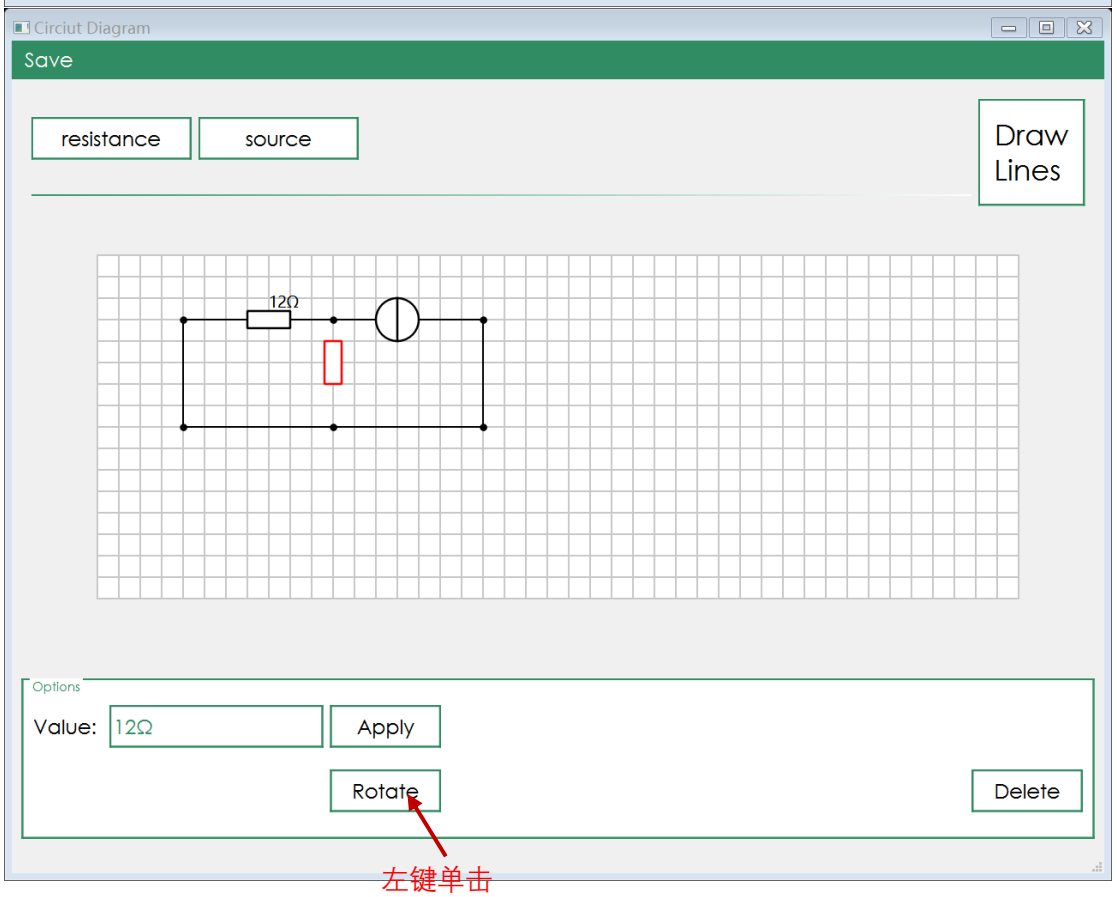


（6）更改元件方向

①左键单击选择元件，元件高亮显示。

②点击“Rotate”按钮，改变元件方向，连线和标记值会自动更改位置。


Circiut Diagram
Save
resistance
source
Draw Lines
12Ω
左键单击
Options
Value:
12Ω
Apply
Rotate
Delete


Circiut Diagram
Save
resistance
source
Draw Lines
12Ω
Options
Value:
12Ω
Apply
Rotate
Delete
左键单击

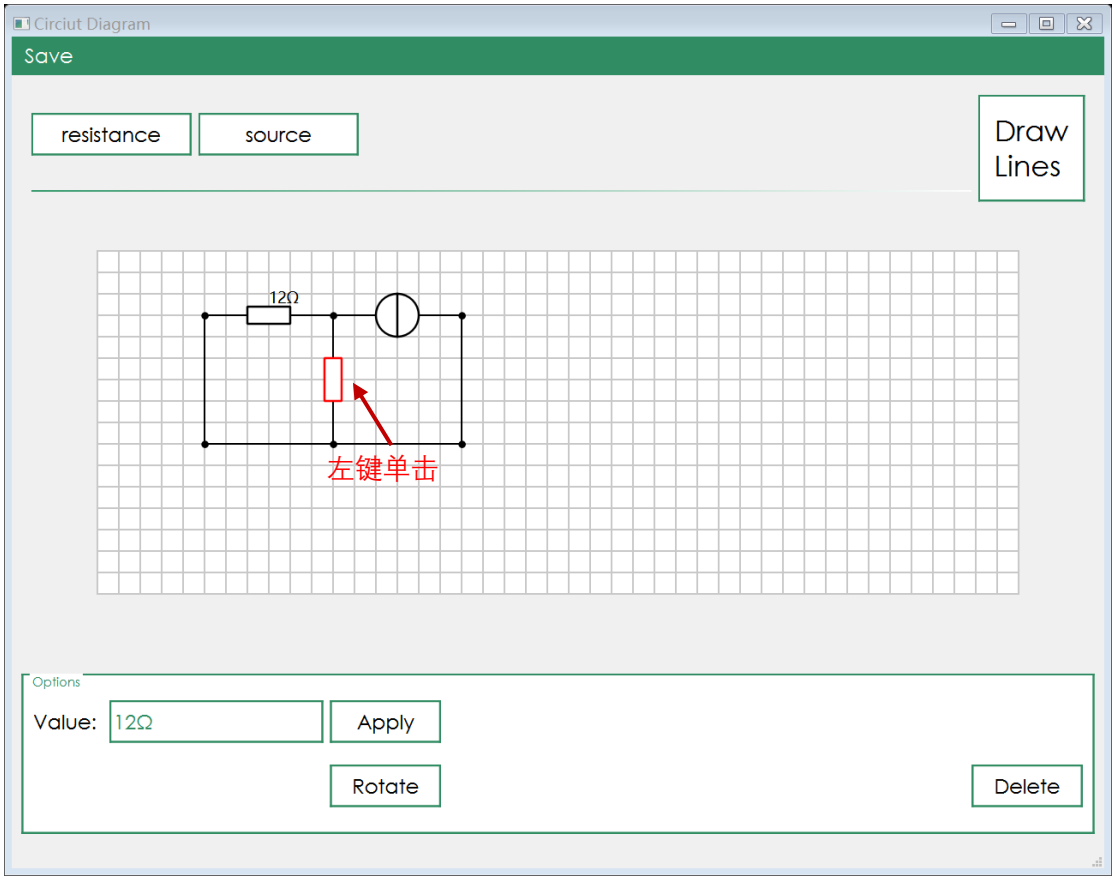
(7)删除元件

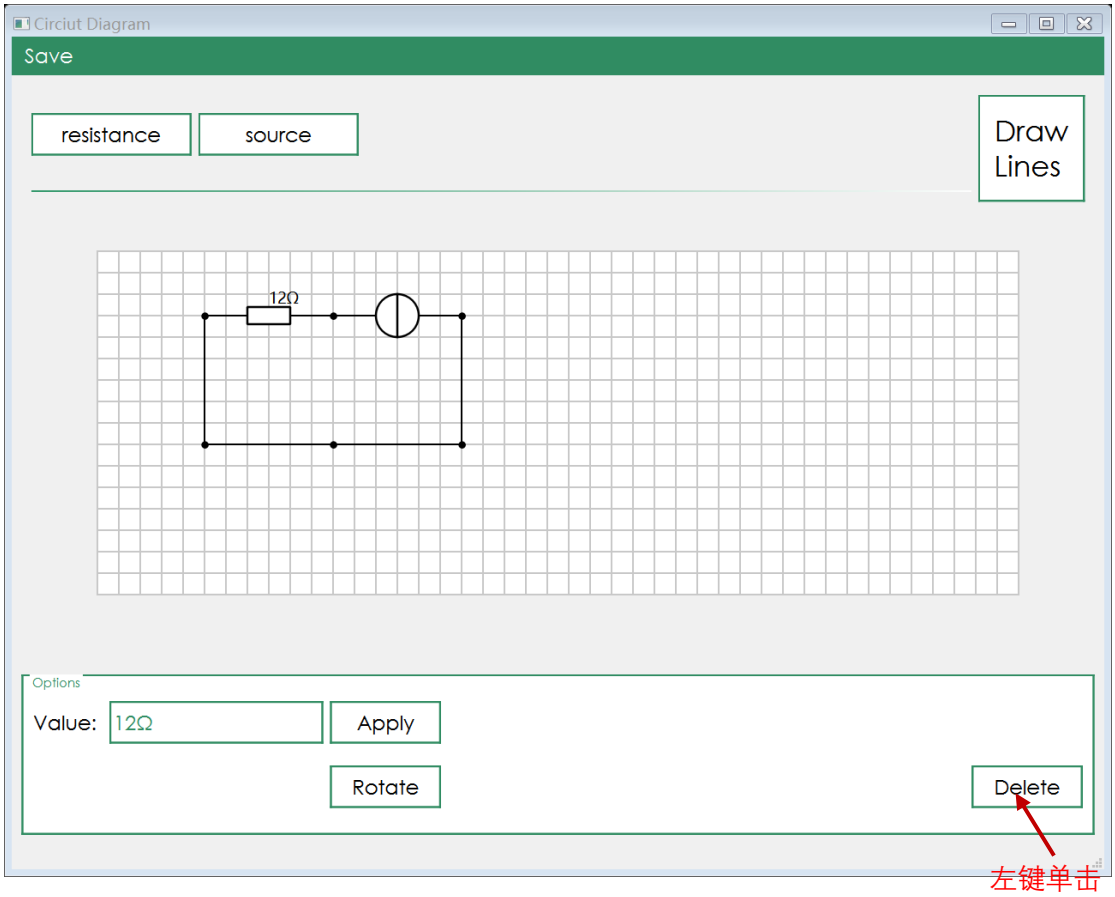
①左键单击选择元件，元件高亮显示。

②点击“Delete”按钮，删除元件，相关的连线和标记值会自动被删除。

①左键单击选择元件，元件高亮显示。

②点击“Rotate”按钮，改变元件方向，连线和标记值会自动更改位置。

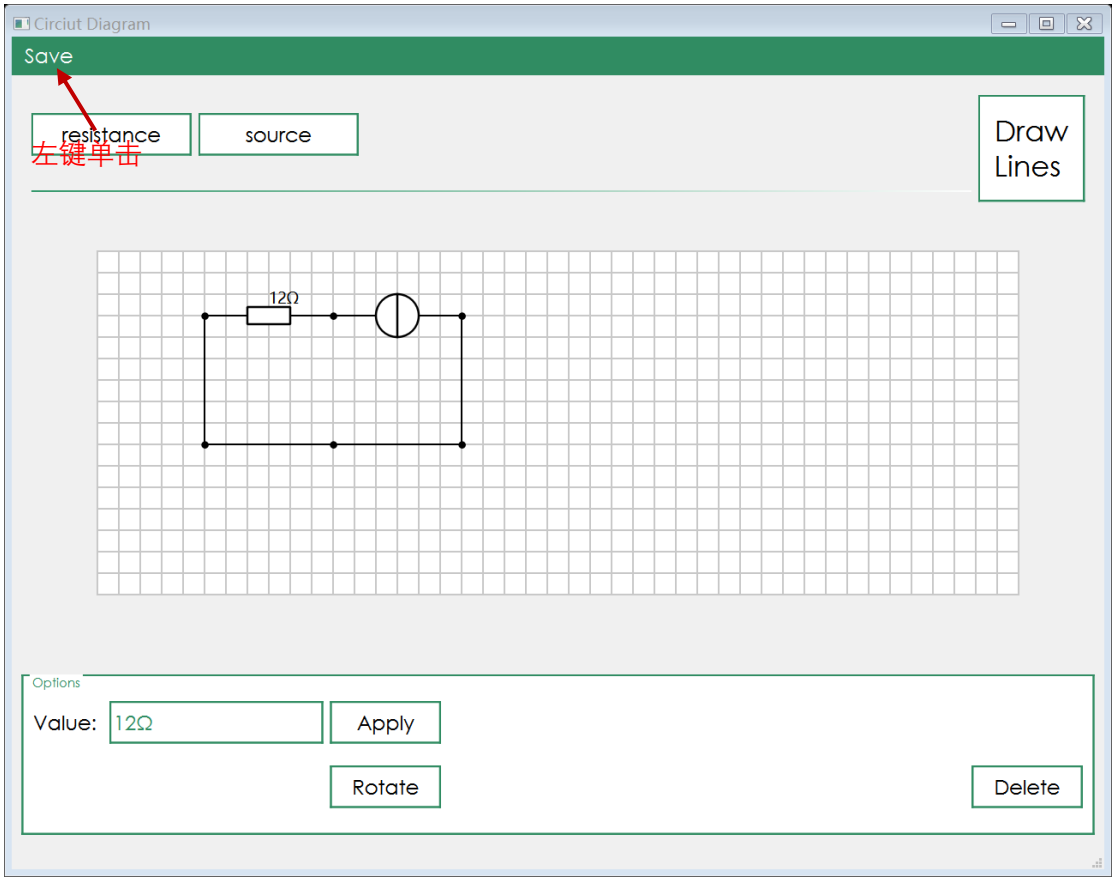


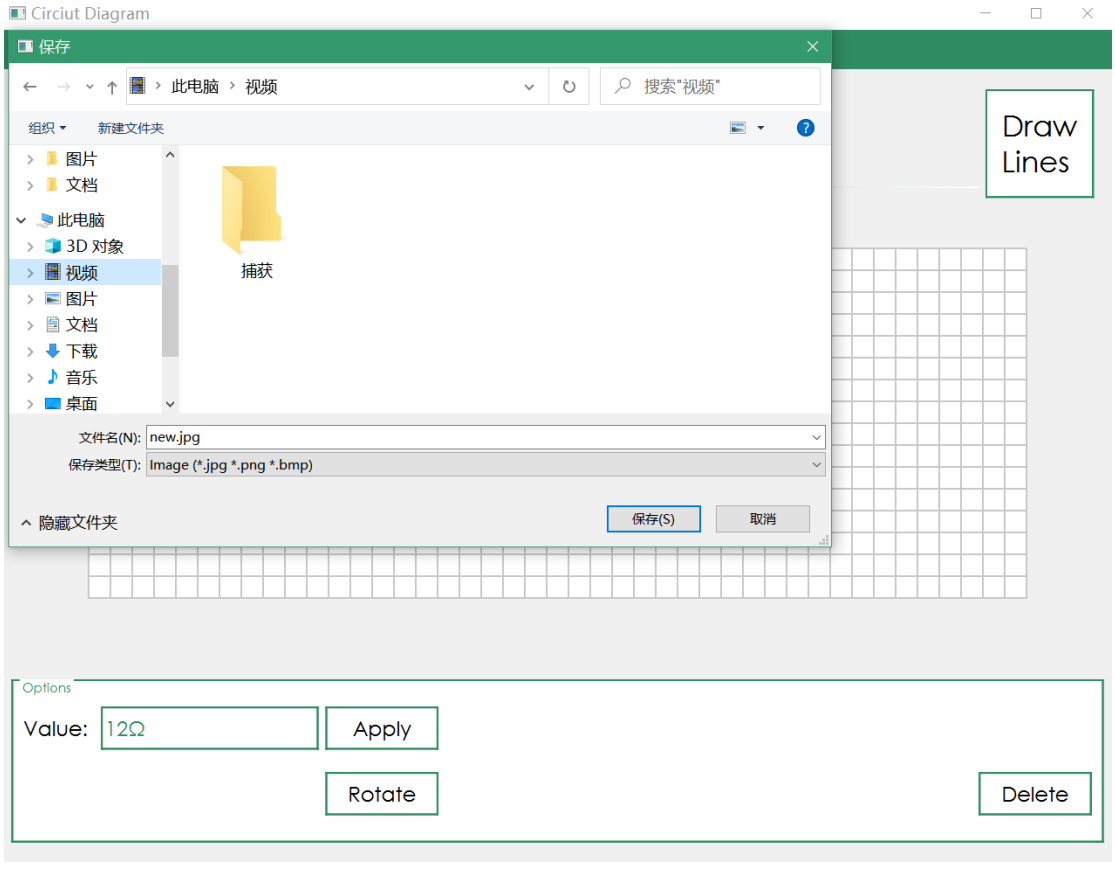


（8）保存

①左键单击菜单栏“Save”按钮

②选择文件夹进行保存


Circiut Diagram
Save
resistance
source
左键单击
Draw Lines
12Ω
Options
Value:
12Ω
Apply
Rotate
Delete


Circiut Diagram
保存
此电脑 › 视频
搜索"视频"
组织
新建文件夹
图片
文档
此电脑
3D 对象
视频
图片
文档
下载
音乐
桌面
捕获
文件名(N):
new.jpg
保存类型(T):
Image (*.jpg *.png *.bmp)
隐藏文件夹
保存(S)
取消
Draw Lines
Options
Value:
12Ω
Apply
Rotate
Delete

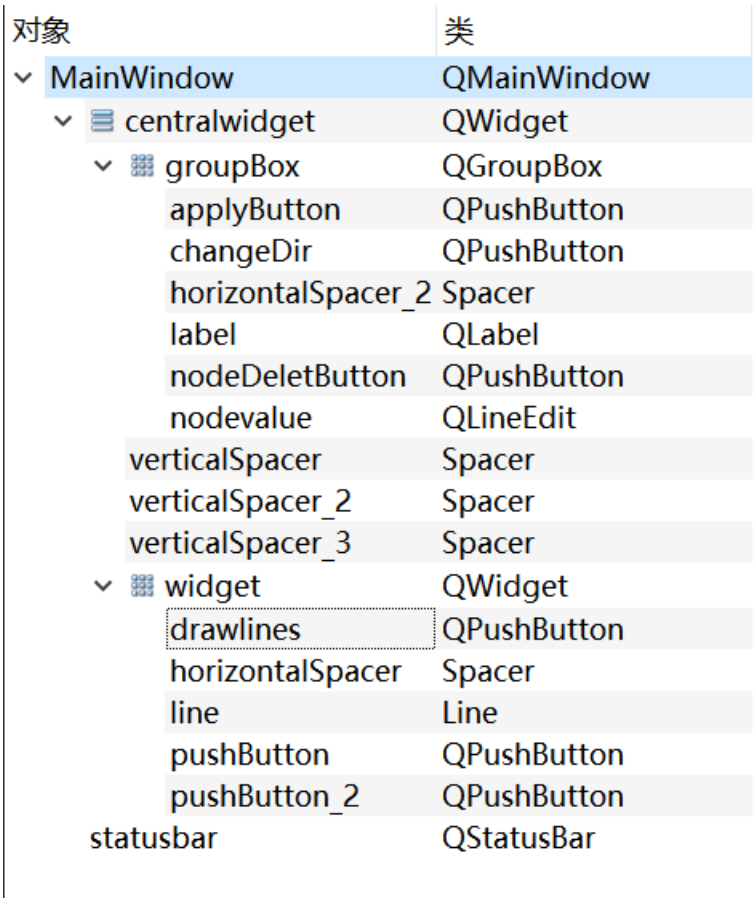
3. 类设计

（1）mainwindow

包括各按键点击事件的处理函数，鼠标事件的处理函数。
具体实现如下图所示：

- MainWindow
  - MainWindow (QWidget * = nullptr)
  - ~MainWindow ()
  - mouseMoveEvent (QMouseEvent *)
  - mousePressEvent (QMouseEvent *)
  - mouseReleaseEvent (QMouseEvent *)
  - paintEvent (QPaintEvent *)
  - SavePic ()
  - on_applyButton_clicked ()
  - on_changeDir_clicked ()
  - on_drawlines_clicked ()
  - on_nodeDeletButton_clicked ()
  - on_nodevalue_textChanged (const QString &)
  - on_pushButton_2_clicked ()
  - on_pushButton_clicked ()
  - ui Ui::MainWindow *

（2）View



（3）Storage

因为元件频繁的删除操作，利用链表进行存储

①元件类 node：

包括元件的相关信息：坐标、大小、编号等

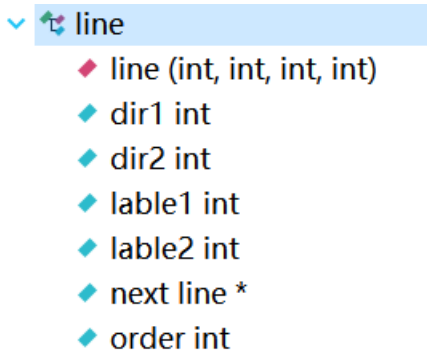
具体实现如下图所示。

- node
- getPos (int, int, int &, int &)
- in (int, int, int)
- isclose (int, int, int, int)
- Rlength const int
- Rwidth const int
- attention bool
- attention_order int
- boardheight int
- boardwidth int
- boardx int
- boardy int
- button_checked bool
- button_type int
- draw_line_mod bool
- eleList MyList<node>
- gap const int
- haveMove bool
- lineList MyList<line>
- line_button_checked bool
- nodesize const int
- nodevaluetext QString
- point_attention bool
- point_attention_order int
- point_dir int
- tem_point_attention_order int
- tem_point_dir int

②线段类 line：

包括线段的相关信息：两端连接的结点编号等

具体实现如下图所示。

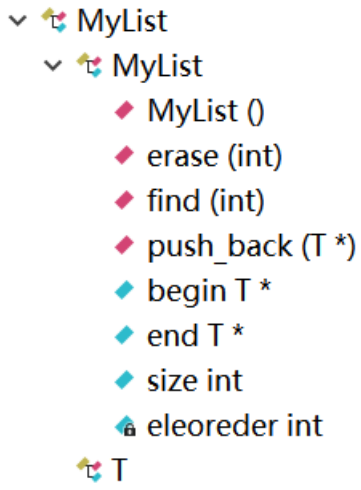


③元件及线段的存储模板类 MyList：

包括链表的相关信息：首尾结点、存储个数等

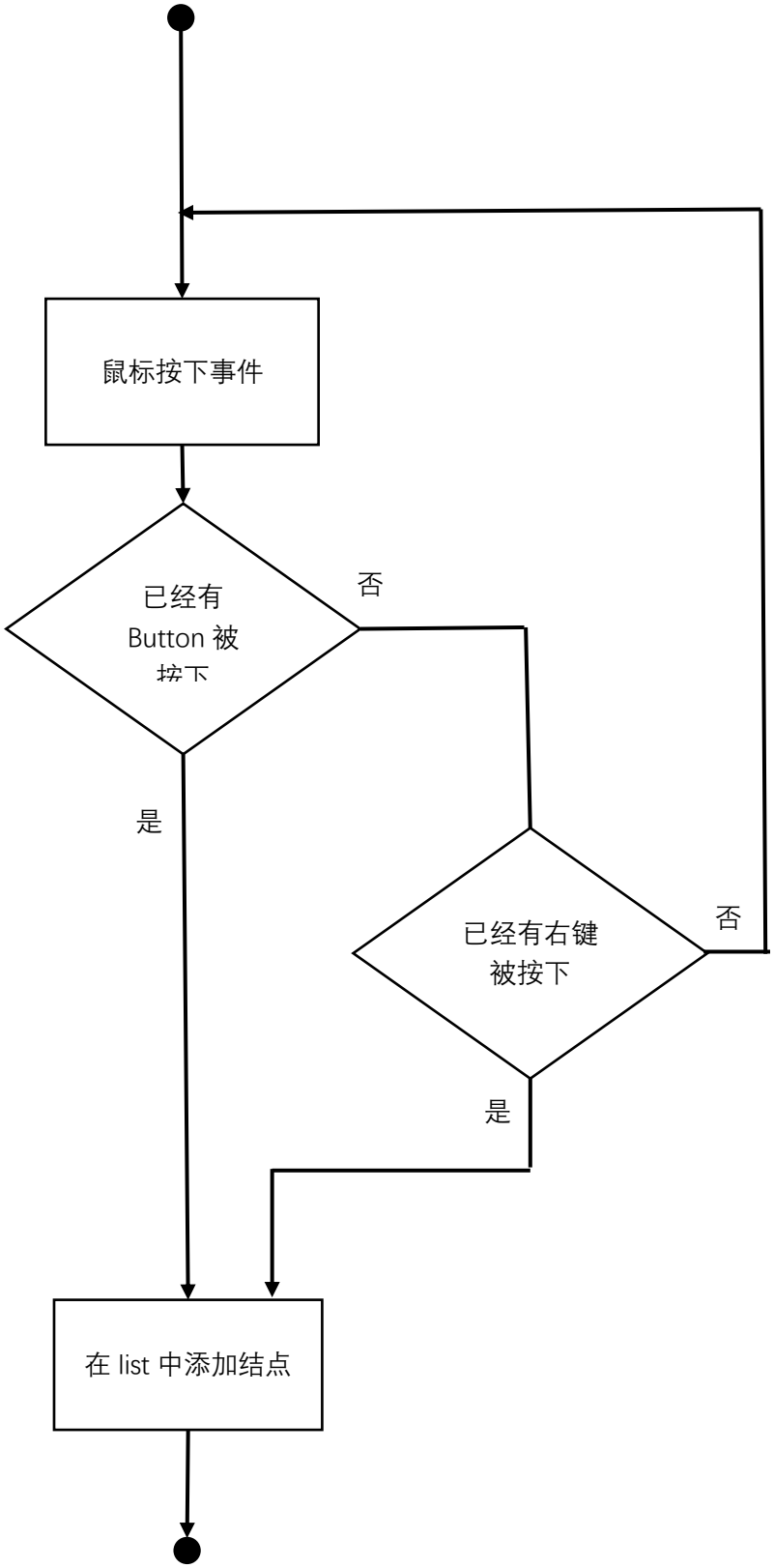
函数包括：尾部插入新结点、删除结点、根据编号查找结点等。

具体实现如下图所示。

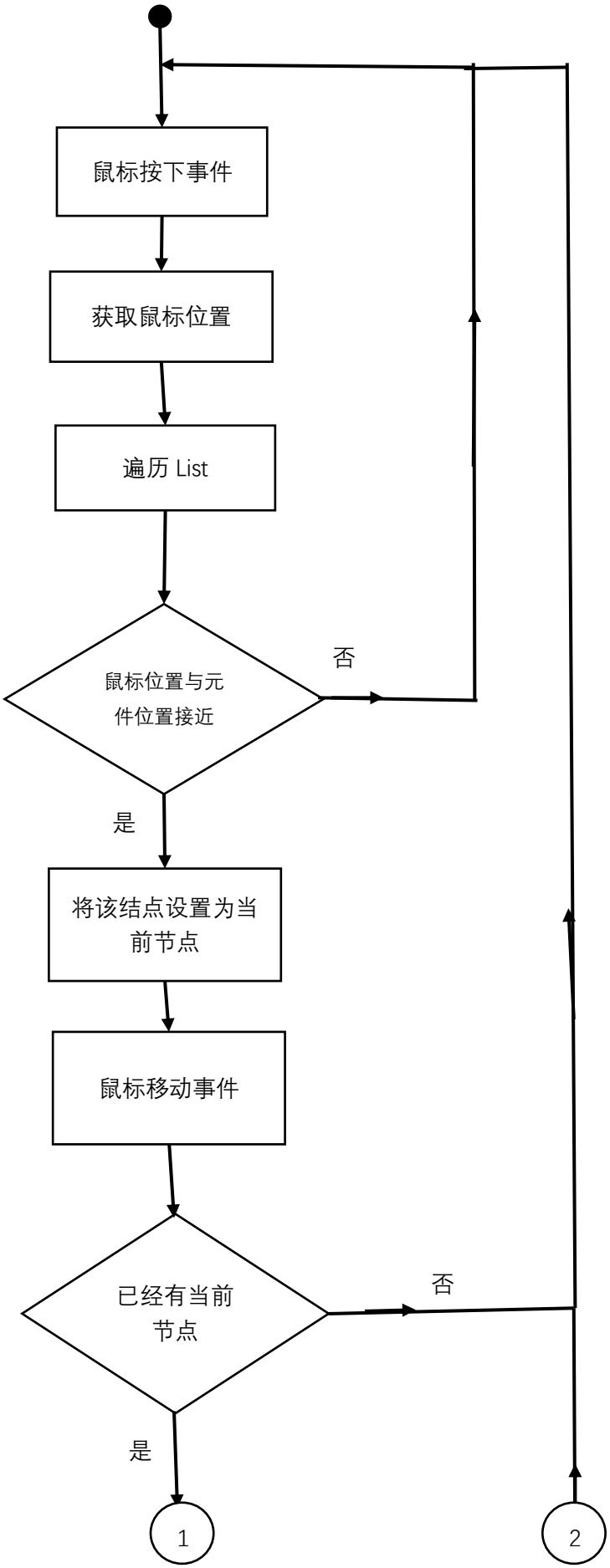


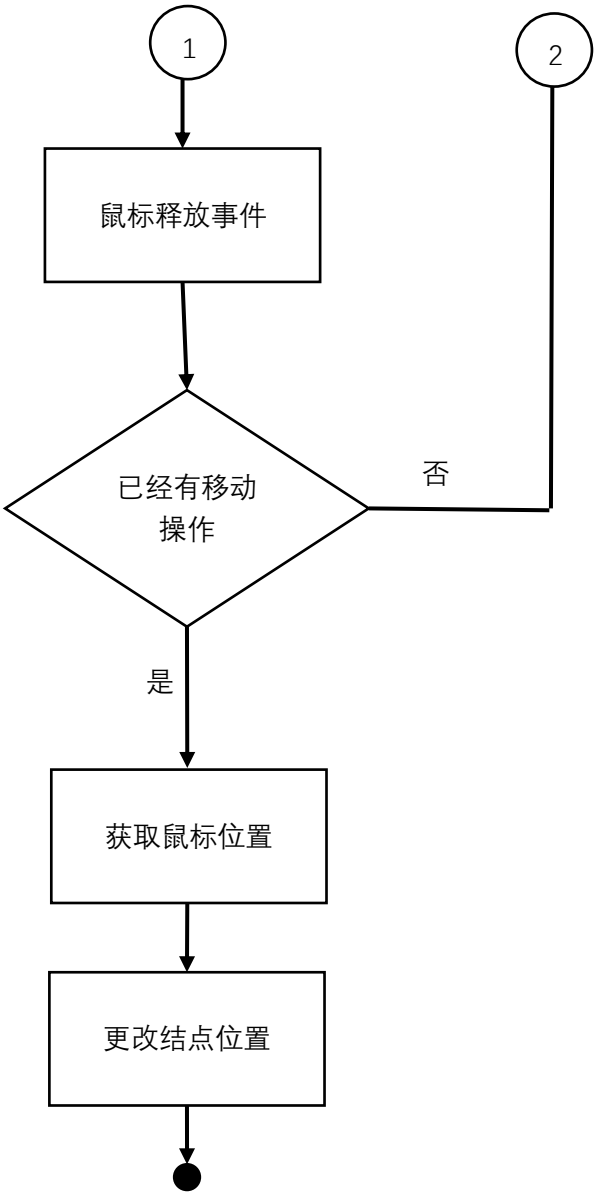
4. 流程设计(算法及技术要点)

（1）新建元件

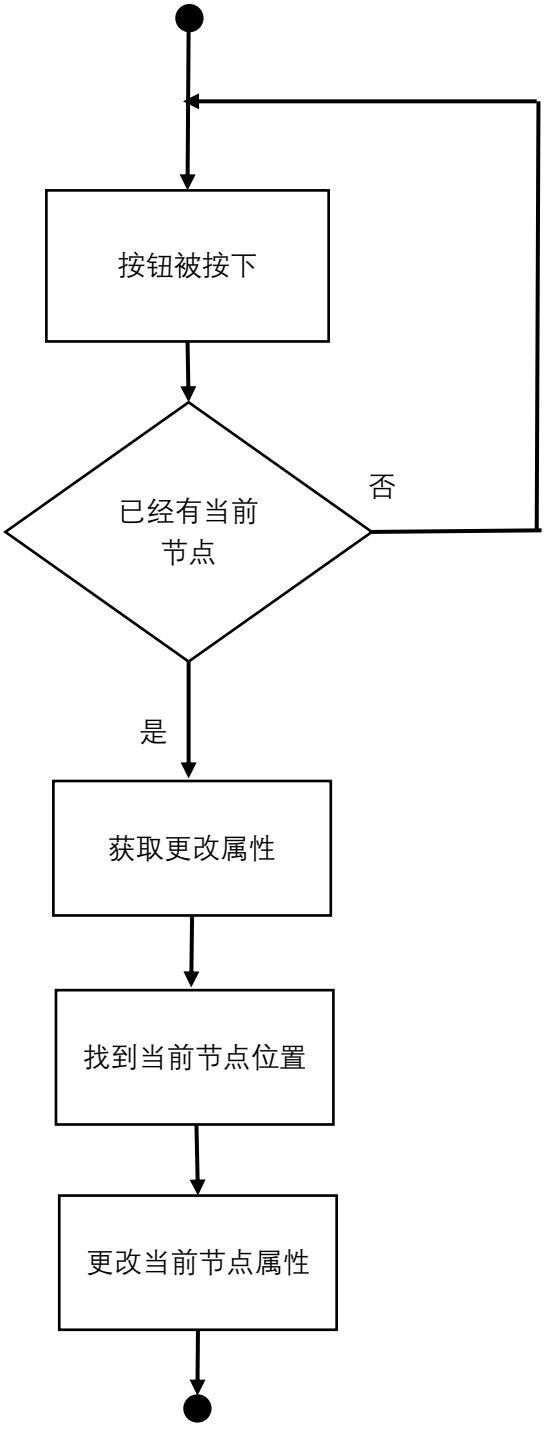


（2）元件移动

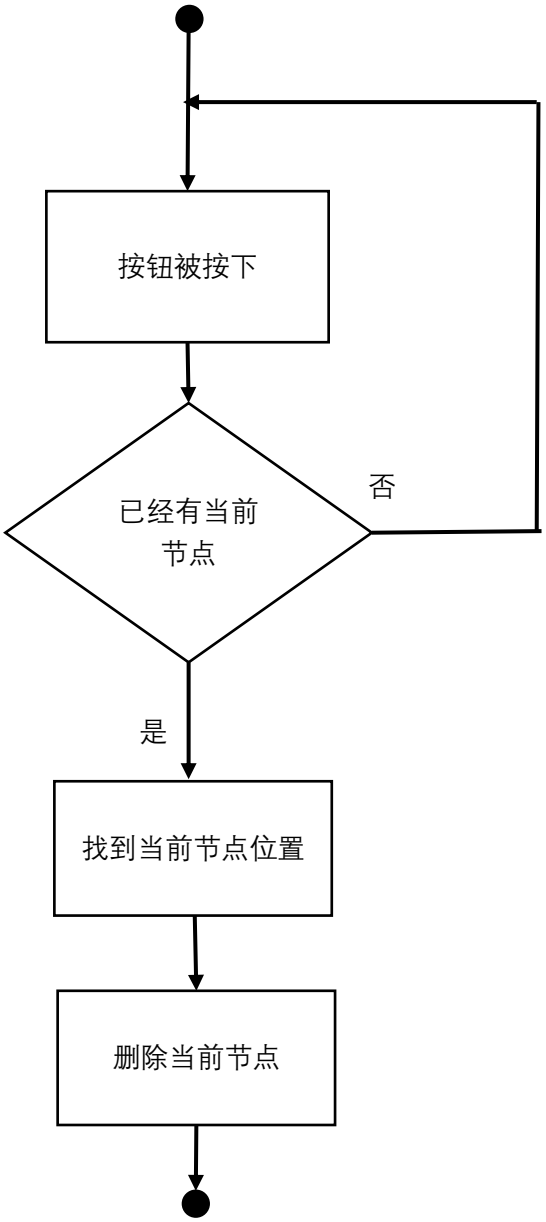

鼠标按下事件
获取鼠标位置
遍历 List
鼠标位置与元件位置接近
否
是
将该结点设置为当前节点
鼠标移动事件
已经有当前节点
否
是
1
2


1
2
鼠标释放事件
已经有移动
操作
否
是
获取鼠标位置
更改结点位置

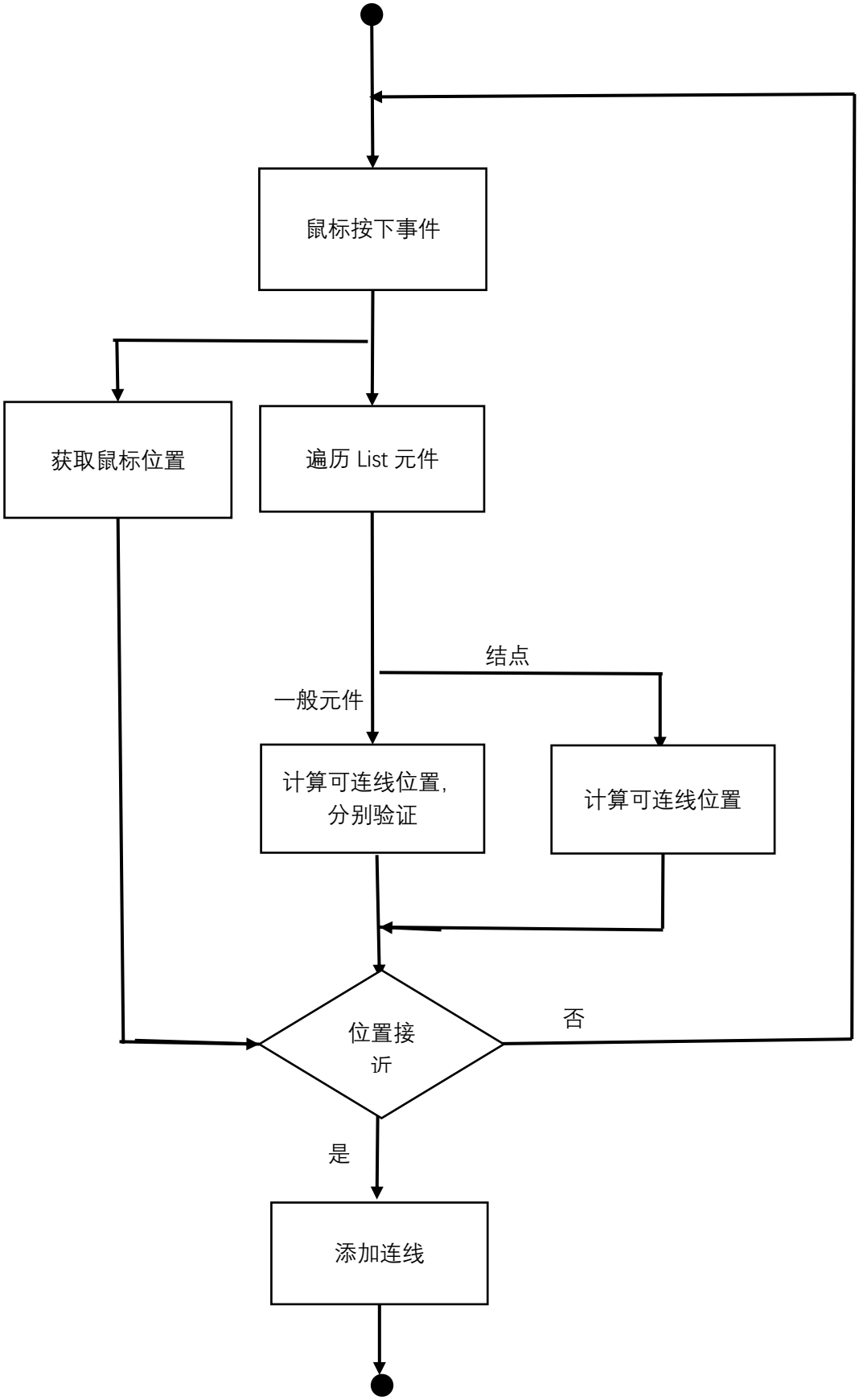
（3）元件属性更改



（4）元件删除



（5）新建线条


鼠标按下事件
获取鼠标位置
遍历 List 元件
结点
一般元件
计算可连线位置，
分别验证
计算可连线位置
位置接
近
否
是
添加连线

五、单元测试

测试案例

表 1：测试案例

输入	输出	目的
生成元件	正确显示	元件显示在网格内
拖动元件	正确显示	元件位置正确变动，并且相关连线正确变动
生成连线	正确显示	连线到元件的一边或节点正中间
删除元件	正确显示	元件消失，相关连线消失
添加元件属性	正确显示	元件旁边出现标注
改变元件方向	正确显示	元件方向改变，相关连线的连线位置自动调整，标注值的位置自动调整

测试结果

通过测试。

六、收获

1.基础算法是函数库还是自建

在存储原件集合时，开始选用了 QLIST 类库，但由于并不是很熟悉，导致如对集合元素序号调整等一些操作产生误解，造成 bug。在估计了深入学习类库与自己简单实现的代价后，快速用自己实现的链表替代类库。通过项目，我更加深入的体会到了类库的作用与代价。

2. 程序设计与变更

对于不同元件，因为实现方式类似，数目不多，所以开始未进行类继承的框架设计，程序可以运行，用课上所学的“基类-继承”等知识来设计反而觉得很麻烦。然而，当需要对不同部件添加不同属性时，在由此方式实现的程序，修改变

得相当困难。

3.项目可行性分析的重要性

在项目开始之前，可行性分析非常重要，如果在开始实现后发现问题，就会付出更多代价。