《计算机图形学》系统设计3月进展报告

141242068-欧先飞

（南京大学 计算机科学与技术系）

1. 实验进度描述

|  |  |
| --- | --- |
| 算法 | 实现状态 |
| 绘制直线的bresenham算法 | 已实现 |
| 绘制圆的中点圆算法 | 已实现 |
| 绘制椭圆的中点椭圆算法 | 已实现 |
| 旋转算法 | 已实现 |
| 平移算法 | 已实现 |

1. 已完成算法原理介绍
2. 用于画直线的bresenham算法

对于给定的起点和终点，这里不失一般性可以假设，，且直线的斜率小于1，斜率大于1和其它象限的线段可以通过坐标变换获得。

Bresenham算法的重点是绘制当前点之后，下一个点是选择还是选择点。Bresenham算法的决策终点便是，如果原直线在的坐标大于，那么选择点，反之选择点。

而上述决策最终包含在代数算式中：

记两点所定的直线的方程式为，那么在处直线的纵坐标为，其与当前横坐标下的两个候选点的距离分别为：、，两个距离的差分为，又，代入可得，进一步令，则当时应选择点，反之应选择点。

上面已经给出了点的决策算式，下面将给出这个算式的叠加算法，对与进行差分，可以得到（注：这里的是指下一个点的坐标，区别于前面提到的），由是便可以得出整个bresenham算法的流程。

由于上述算法只能处理第一象限内斜率小于等于1的直线，所以对于其它情况需要稍作变换，对于斜率大于1的直线可以将与的角色互掉，这样原直线以为基之后斜率便小于1了，然后即可使用上述的bresenham算法，对于第四象限斜率小于-1的直线，将原推导式中替换成即可，而对于第四象限斜率大于-1的直线，可以先将其与的角色互掉，然后将起点重点的角色互掉，最终便可以变换到第四象限斜率小于1的情况，然后直接套用即可。

而对于第二、三象限的直线，可以将其起点终点角色互换，变换到第一、四象限，然后套用上述方法即可。

1. 用于绘制圆的中点圆算法

对于给定的圆心和半径，这里不失一般性可以假设，，并可以先只绘制圆的八分之一（第一象限直线的上方），其余部分可以通过坐标变换得到。

与绘制直线的bresenham算法类似，中点圆算法的重点是绘制当前点之后，下一个点是选择还是选择点。中点圆算法的决策方法是，如果圆曲线在的坐标大于，那么选择点，反之选择点。

为了方便起见，定义函数，若，则点在园内，反之在圆外。所以对于坐标，若，则选择点，反之选择点。

令决策函数，逐差可得（注：这里的是指对应横坐标处实际选择点的纵坐标，区别于前面使用的，如果，那么，选择点，如果，那么，选择点。

由上述流程可以获得第一象限上方的八分之一圆，第一象限下方的八分之一圆可以通过将上方八分之一圆所有点的和坐标互掉，然后再将得到的所有店依次沿坐标和坐标作对称变换。

3、绘制椭圆的中点椭圆算法

中点椭圆算法的思想基本与中点圆算法一致：即椭圆曲线与下一个坐标的交点离哪个候选点更近一点。但由于椭圆并不依直线对称，所以在第一项限的上半部分和下半部分之间不能通过坐标变换得到对方。这里不妨将上半部分（斜率大于等于-1的部分）称为区域Ⅰ，下半部分（斜率小于-1的部分）称为区域Ⅱ。同时不失一般性的可以设椭圆中心坐标为，椭圆的方向半轴为，方向半轴为。

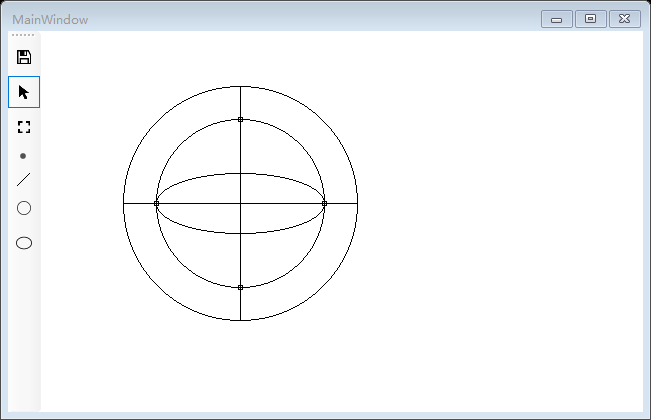
设，则当时可以判定点落在椭圆内部，当当时可以判定点落在椭圆上，当时可以判定点落在椭圆外部。同时为了判定当前点时在区域Ⅰ还是区域Ⅱ，需要实时检测当前的斜率，斜率到达临界点时满足。在斜率越过临界点之后，需要将采样方式由沿轴采样变更为沿轴采样。

记区域Ⅰ的决策函数，则当时，选择点，反之应选择点。

记区域Ⅱ的决策函数，则当时，应选择点，反之，应选择点。

由上述流程便可以获得第一个象限内椭圆的各像素点的计算方法，然后再对这些点依次沿轴和轴作对称变换，便可以获得其它三个象限的像素点。

1. 实现效果截图



1. 程序具备的功能
2. 通过点击左侧的图标按钮，可以在右侧的画板上绘制相应的图形。
3. 绘制说明

当右侧面板处于绘制图形状态时，正在绘制的图形会处于选中状态（端点上会有小矩形），绘制结束后，图形选中状态消失，可以通过move按钮重新选中图形。

1. 绘制点

先点击左侧菜单栏的图标，确认该图标被选中后可以在右侧面板上单击绘制点



1. 绘制直线

先点击左侧菜单栏的图标，确认该图标被选中后，在右侧面板上先按下鼠标，得到直线的一个端点，然后摁住鼠标不妨，移动鼠标，可看到直线随鼠标变化，最后松开鼠标，得到直线的另一个端点，直线绘制完毕。



1. 绘制圆

先点击左侧菜单栏的图标，确认被选中后，在右侧面板上先按下鼠标，获得圆心，然后在摁住鼠标不放的状态下拖动鼠标，获得圆的边界点，最后松开鼠标，圆绘制完毕。



1. 绘制椭圆

先点击左侧菜单栏的图标，确认被选中后，在右侧面板上先按下鼠标，获得椭圆中心，然后拖动鼠标，鼠标到椭圆中心的x坐标之差作为椭圆的x半轴，y坐标之差作为椭圆的y半轴，最后松开鼠标椭圆绘制完毕。



1. 点击画板上的图形可以重新选中该图形。

先点击左侧菜单栏的图标，然后即可在右侧面板上选择已绘制的图形。



对于直线，点击线段可选中，对于圆，点击边界线可选中，对于椭圆，点击椭圆内部可选中。

1. 鼠标落下状态时拖动鼠标可以移动被选中的图形。

先点击左侧菜单栏的C:\Users\欧先飞\AppData\Local\Microsoft\Windows\INetCache\Content.Word\cursor.png图标，然后在目标图形上按下鼠标，此时拖动鼠标图形即会跟随鼠标移动，松开鼠标，图形会移动至新的位置。

1. 按退格键可以从画布上删除被选中的图形。

先按2的方法选中该图形，然后按下退格键即可删除该图形。

1. 支持重新调整被选中的图形。

先点击左侧菜单栏的图标，然后点击目标图形并拖动鼠标可以改变目标图形的大小，形状。比如摁住线段的一个端点并拖动鼠标，该直线的这个端点会跟随鼠标移动（另一个端点维持不变）。

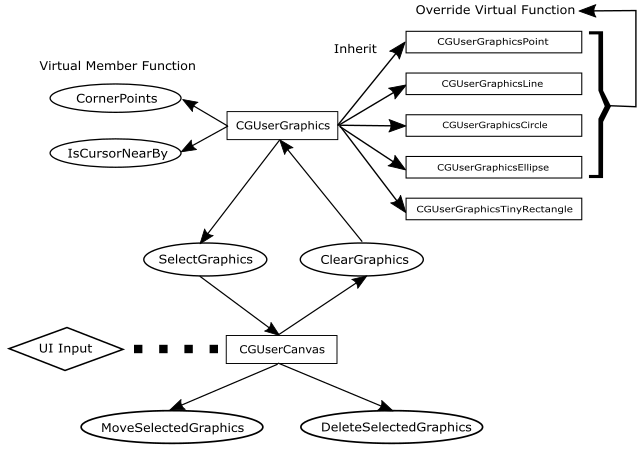


1. 支持保存到文件。

点击左侧菜单栏的图标，在弹出的保存文件对话框中输入文件名，然后即可保存。



1. 系统框架设计详述
2. 类关系视图（方框代表类、椭圆代表方法）



1. CGUserGraphics类方法描述

CGUserGraphics是用户自定义图形的基类，其目前为止定义了4个方法：CornerPoints、IsCursorNearBy、DrawTagPoints、ClearTagPoints，其中前两个方法是虚函数，并期望在子类中被复写。

1）CornerPoints用于获取该图形的标记点，举个例子，对于直线而言，其标记点是两个端点，而对于圆来说，其标记点是四个端点（与坐标轴的交点）。通过端点可以在视觉上给予用户以反馈，让用户知道自己选中了该图形。

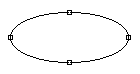
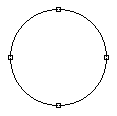
2）IsCursorNearBy用于判断当前鼠标的位置是否落在该图形上面，对于圆而言，只需要鼠标的位置与圆边界的距离小于4就可以判定鼠标落在了该图形上，由是便可以将该图形转换为选中的状态。

1. CGUserGraphics子类描述

CGUserGraphics是全部用户定义图形的父类，其截至目前为止共有5个子类：CGUserGraphicsPoint、CGUserGraphicsLine、CGUserGraphicsCircle、CGUserGraphicsEllipse、CGUserGraphicsTinyRectangle，其中前四个图形是依照课程中的算法实现的，最后一个CGUserGraphicsTinyRectangle区别于其它的子类，它只用于绘制标记点。

1. CGUserCanvas类功能描述

1）SetGraphicsSelected用于绘制该图形的标记点，用以在视觉上反馈用户该点已被选中，下图是各个图形被选中时的状态：



2）ClearGraphicsSelected用于在用户级画布上清除标记点，用以标注该图形目前处在未被选中的状态，一般当用户点击其他位置时原来的图形会被清除标记点。

1. CGUserGraphics与CGUserCanvas的关系

CGUserCanvas是工程自定义的画布，通过Select操作可以将一个继承自CGUserGraphics的子类选入画布，等到下一次刷新时被选入的图形会被绘制在窗口中，举个简单的选入操作的例子：



通过选入操作被绘制在画布上的图形不会自动消失，除非进行Clear操作：



同时为了避免clear操作清除掉与其它图形共有的部分，CGUserCanvas为每一个像素点增加了引用计数，只有当某一个像素点的引用计数到0时，该点的像素值才会真正被清除。