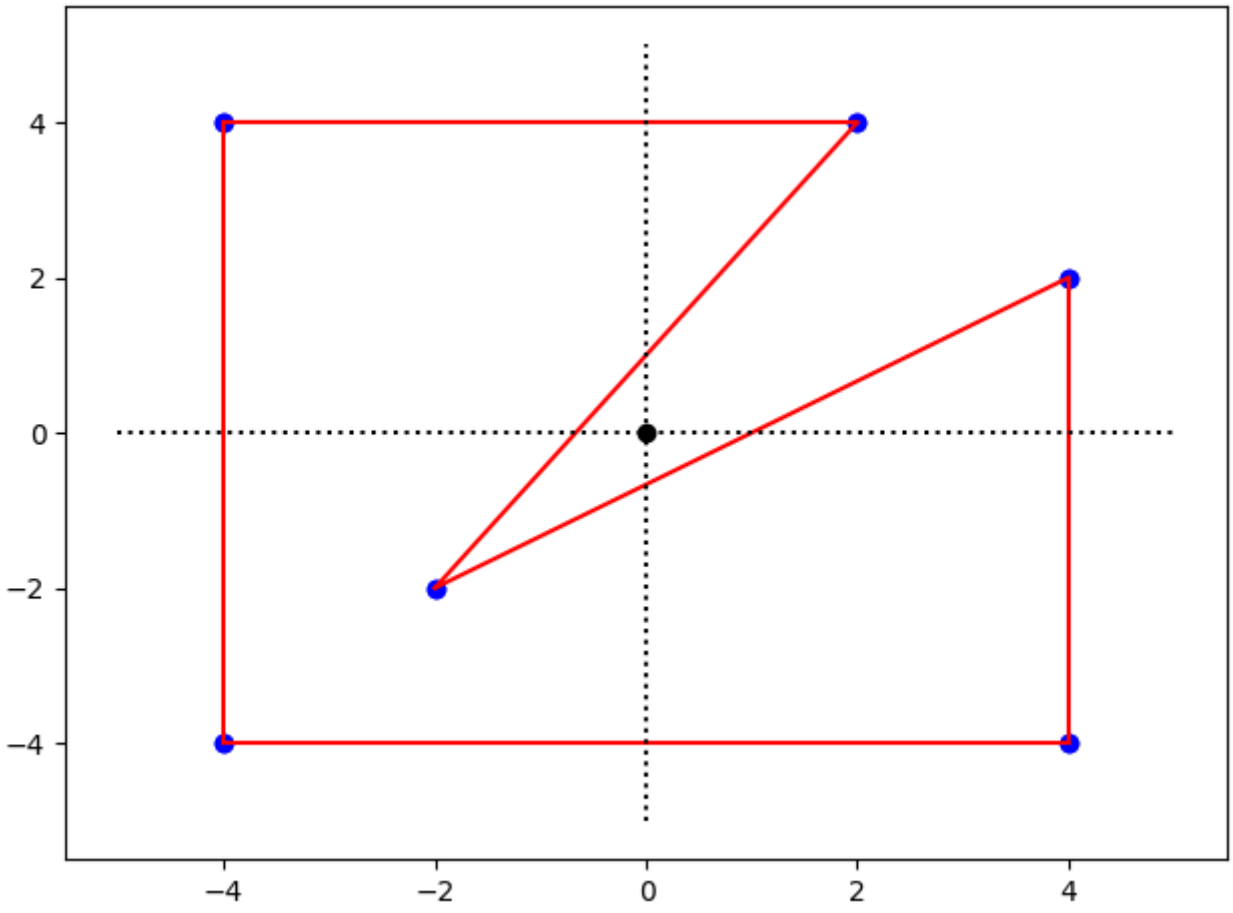
计算机图形学第四次作业与思考题

1953348 叶栩冰

1. 象限法分析

象限法在方法介绍上虽说是判别每条边对应该点的单位圆上面的弧度之和，但是其实质是以所判断点建立直角坐标系，判断多边形的每个边的变化象限范围，依次将变化值求和来根据最后的值来判断点和多边形的相对位置。

在老师初步讲解的阶段并没有讲到如何细节处理跨两象限的边时，我对下图所示情况的处理方法有所疑问。

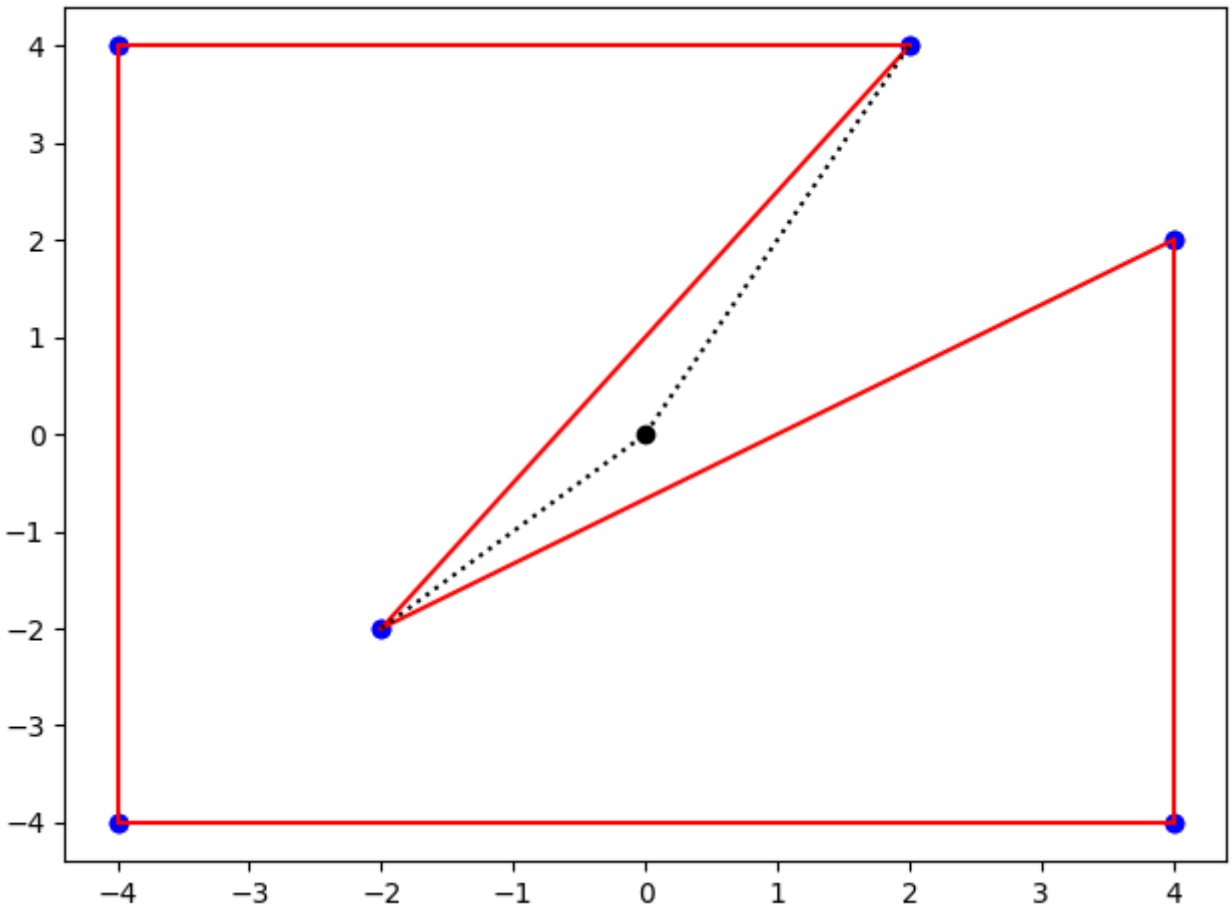


但是随机提出了处理该情况的代数步长公式。

```
def calChange(x, y):
    if y[1] * x[0] - x[1] * y[0] > 0:
        return 1
    else:
        return -1
```

这里用1代表pi以减少计算量，也可以用2来表示这样就完全杜绝了浮点数运算。

该方法的本质是比较下图中两虚线斜率的大小以及增减情况，这实质上也是比较对应边和原点的位置关系即截距的正负。



该方法的计算量及复杂度是相对较低的，那么存在值得去分析的问题就只剩下所选点及坐标轴与多边形顶点的位置关系问题了。

```
def cal(x, y):
    ans = 0
    for i in range(len(x)):
        Loc1 = tellLocation(x[i][0], y[i][0])
        Loc2 = tellLocation(x[i][1], y[i][1])
        if abs(Loc1 - Loc2) == 0:
            pass
        elif abs(Loc1 - Loc2) == 2:
```

```
            ans = ans + calChange(x[i], y[i])
        elif Loc1 - Loc2 == 1 or Loc1 - Loc2 == -3:
            ans = ans - 0.5
        else:
            ans = ans + 0.5
    assert (ans == 0 or ans == 1 or ans == 2)
    explain = {
        0: "在外部",
        1: "在边界上",
        2: "在内部"
    }
    return ans, explain.get(ans)
```

在我去复现算法的过程中，如上述代码块所示，由于两个点的位置变化情况存在多种可能，因此常人的思路的第一部操作是去将点的位置转化为象限，然后去判断象限的变化情况就会十分简单，但是会存在多边形顶点在坐标轴上的情况。因此只需要做一个规定即可，将两条坐标轴的四个部分分给四个象限即可，即x+,y+,x-,y-分别归属一二三四象限。

至于点在边上的情况，老师在ppt内也作了说明，当排除所验证点为多边形顶点情况下，只有发生跨两象限的情况时，才会出现点在多边形边上的可能。因此只需要在代数步长公式中增加一个case为0的情况，并立刻抛出，即可解决。

> 详细代码及实现过程见附件。

2. 射线法（三维）

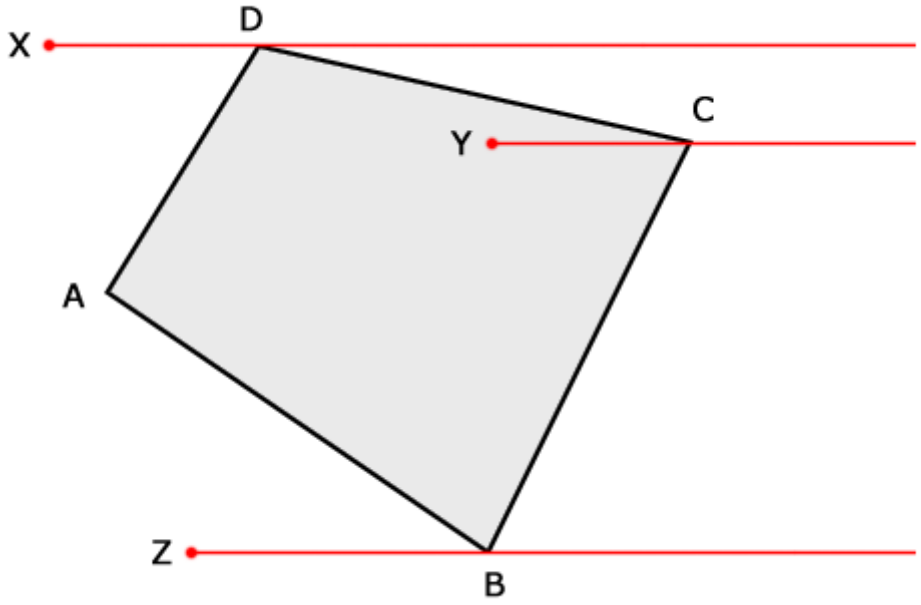
首先来看一下上课讲过的二维空间内的射线法。

通过判断射线穿过多边形边的次数的奇偶去判断点在多边形内还是外。其实质思想是，任意一条直线穿过多边形的次数一定为偶数次（假设不过顶点），那么射线穿多边形的次数的奇偶仅仅取决于第一次穿过多边形为穿入or穿出。入偶出奇。

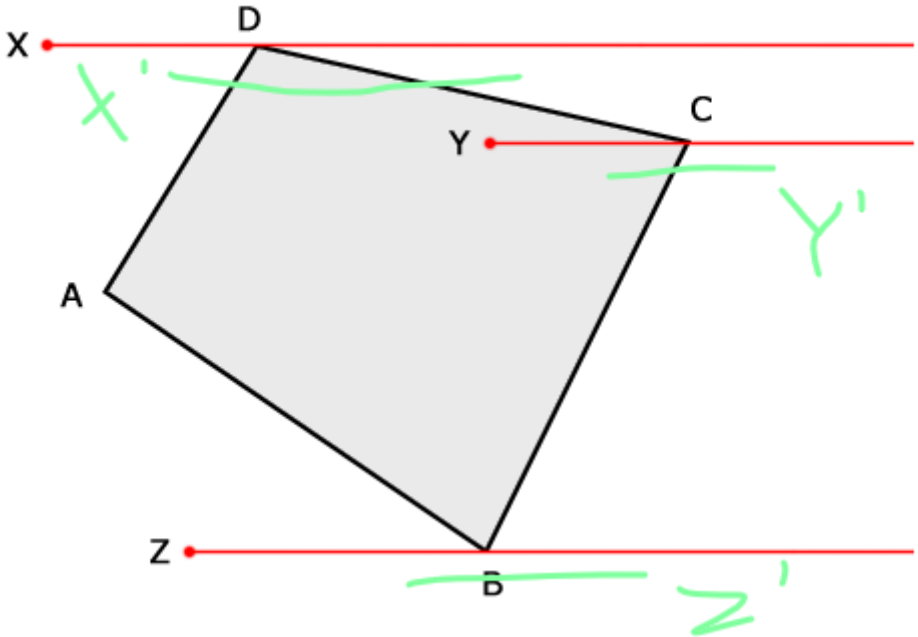
下面来讨论一下二维空间使用射线法时存在的特殊情况：

1. 点在多边形边上
2. 点是多边形顶点
3. 射线经过顶点
4. 射线经过边

以上几点是我在上课过程中疑惑的tips，那么想要处理这些情况就要从射线法计算是否穿过某条边的计算方法出发。射线穿过线段实质是线段的两个端点分别在射线的两侧，因此我们要对端点和穿过射线的位置关系进行明确的规定。



如上图所示，我们规定端点属于所穿过直线的上方，如下图所示。



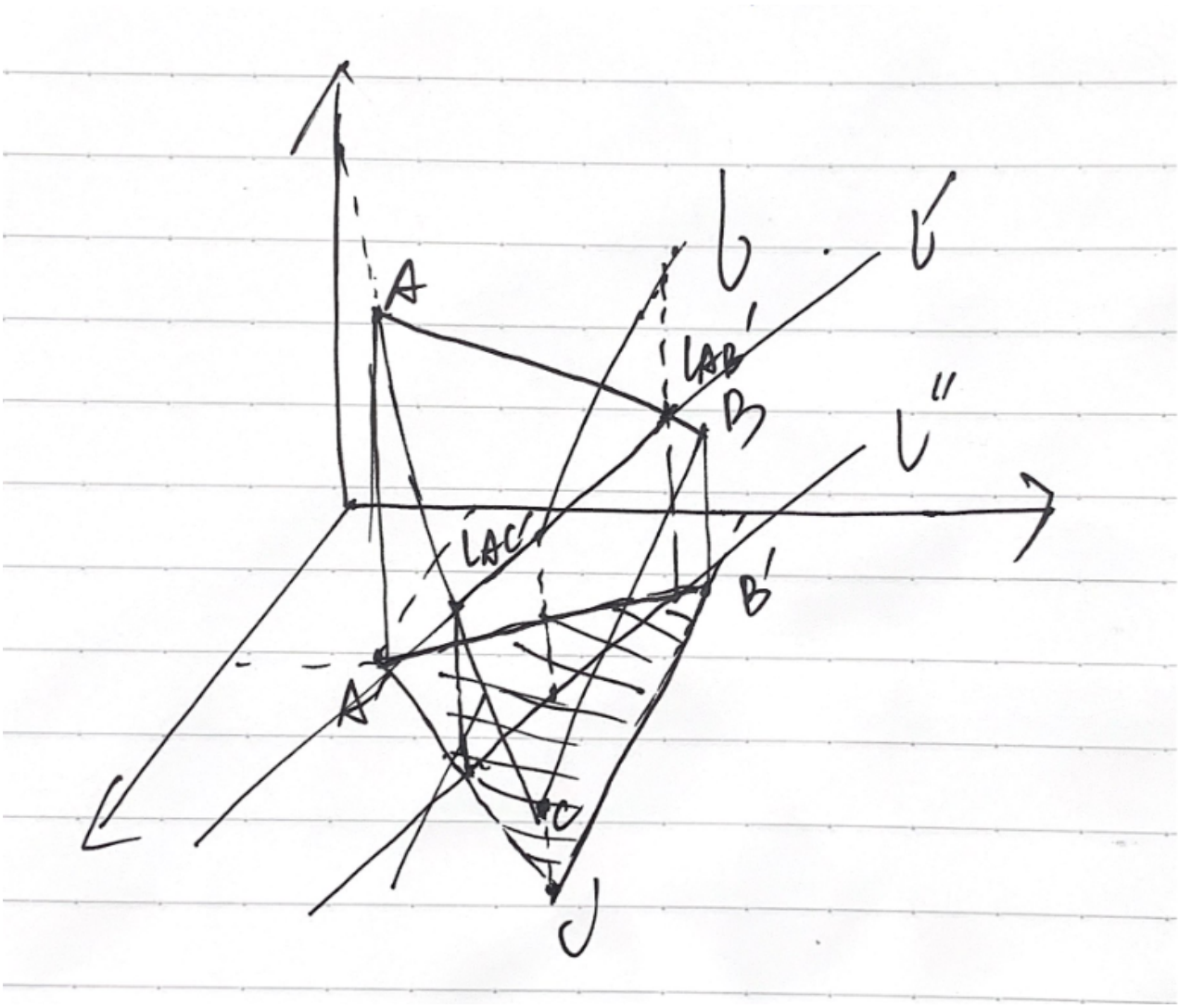
那么X穿过两次，Y穿过一次，Z并没有穿过多边形。那么上述问题就都迎刃而解。

下面来到射线法求解点是否在三维体内部：

我们将这个三维体分为两种情况，一种是规则三维体即多面体，在一种是不规则三维体（可能含有曲面或者并非凸包），由于三维的计算量要比二维大得多，因此提供一个优化思路：可以先求解该三维体的最小包围长方体，先计算点是否在这个最小保卫长方体内，如果不在的话就不可能在三维体内部，即无需进行计算。

然后我们来思考射线法在三维体内的使用，第一种情况即规则多面体，其实质仍是求射线是否与该多面体的各个平面相交，因此就引申成射线与区域平面的相交问题。那么我们可以参考二维的思路。在二维平面中，我们是判断线段两端点是否在射线的异侧来判断该射线是否穿过线段，那么在三维空间中我们依旧可以将其联想。

如下图所示，l为所求射线，ABC为所求相交平面，那么我们如何不去求线面交点就确定这条线是否与这个平面相交呢？我们可以在线与面的交点附近选取两电，看着两点是否在面的异侧，为了规范化这两点的选取并且能够保障交点是在区域平面而非整平面内，我们选取lAB'和lAC‘两点，这两个点是射线l和平面ABC分别在xoy平面上投影所得到的区域和直线，在二维空间内找到他们的交点的计算复杂度就要远远小于三维空间。找到交点后验证其z坐标即可。



上述算法将三维问题简化成了二维问题，降低了复杂度，针对于特殊情况和非规则化多面体的情况在此不探讨。