LIGHT-2D 几何光学工具说明

17307110177 魏雨轩 物理学系

摘要：此工具提供了一个自由的几何光学平台，允许用户自由绘制各种形状的透镜，平面镜，并且对这些器件进行旋转平移，自由搭建几何光学系统。用户可以加入任意多的光源或者器件，并且查看光在系统中折射反射形成更多束光的现象，而且光束分支的能量由透明度控制。这个系统可以为几何光学相关的知识点讲解和实验设计，误差分析等提供帮助。

一， 引言

中学老师在讲述光学成像等几何光学内容的时候，由于不能够把实验器材搬到课堂现场展示，学生对成像过程中一些动态变化的过程可能理解不是很深刻，导致在做实验调光路的时候不知道如何操作，如何通过移动透镜使得成像在正确的位置。这个时候需要一个实时的交互的软件，可以模拟搭建光路的过程，可以代替搭建光路的过程，让学生在做实验的时候更得心应手。

又比如在大学生做迈克尔逊干涉仪时，由于半透半反镜的厚度原因，在观察端往往会出现多个光点，这些光电有强有弱；在成像过程中，由于大部分透镜都是球面镜，像会出现一定程度的畸变；在设计一套激光系统的时候，要学会避免很多不需要的反射折射，这会对光学器件和人产生比较大的安全隐患……这些现象在分析原理的时候不被重视，而试验中又确实会出现，对我们的结果产生一定的影响，所以我们需要一个能够严格遵守 snell 定律的系统，能够最真实的模拟出在实验中遇到的所有可能的现象，这对实验前的实验步骤的安排，和实验后的误差分析都有着很大的作用。

所以 LIGHT-2D 几何光学工具就是为了提供一个可交互的，自由搭建的，在每一个界面严格遵守 snell 定律，并且体现多光束在介质中的透射反射，和对应的光强大小的小型软件，只要用浏览器就可以运行。

二， 程序结构

为了实现上述功能，程序主要分为三个部分：**对象部分**,**寻光部分**和**交互部分**

（1） 对象部分

引入五个对象：

1，光对象：light
2，子光对象：childlight
3，透镜对象：lens
4，平面镜对象：mirros

下面来分别构建这些对象。对象主要有两个部分：**对象的属性**，**对象的方法**，而对象的属性中有和对象直接的形态，大小，物理量等在实验中有实际对应的**实体属性**和为了实现交互而引入的**控制属性**。所有对象都是这样的结构构成，使得在之后的平移旋转等交互操作，可以用相同的代码实现对不同对象的控制。

1，光对象：

为了实现一束光在介质表面分成两束不同能量的光的现象，必须把光这个事物分成两层来看待，一层是最开始的光源光，保存了一开始的光源位置和出射方向，而第二层是由这束光生成的所有子光。

所以光对象的实体属性为：

- Type ： ”light”;属性判断，可以对后续的交互有帮助
- length:默认再也没有折射反射后延长的长度
- rgba:颜色和光强，光强用透明度表示
- Begintheta:出射光的角度
- Width:光线宽度
- Pole:极化方向，0,1，表示平行或垂直于屏幕，用于使用菲涅尔定律计算反射光和透射光的光强
- Childlight：子光列表，用来储存有这一束光产生的所有子光。

控制类属性都有锚点(anchor)，可平移(transferable)，可旋转（rotatable）这是光，透镜，平面镜都共有的属性，之后的叙述中不再强调。

对象方法为：

- Addchild, addchilds, deletechild（子光操作）
- Draw（绘图）
- Init（初始化）
- transfer
- Rotate
- Show_info（信息汇总）

其中第一列是对子光对象的操作。初始化是为了在改变光源位置后，新的必须清空原来的子光列表，初始化是必要的。Transfer,rotate,show_info是所有对象共有的方法，用来后续的旋转平移，和信息的展示。

2，子光对象：

实际反射和透射其实都是子光对象。每产生一个光对象，都要生成一个最初的子光对象，继承光对象的所有实体属性，之后每产生一束新的光线都会在子光列表中添加一束继承了光对象实体属性的子光对象。

光的路径等等数据其实都储存在子光中所以，实体属性有：

- path（光路径）
- end（路径末端坐标）
- Theta（路径末端倾角）
- N（路径末端所处折射率）
- Endwidth（末端的宽度）
- Length（继承光对象）

- Widthset（每段路径的宽度，因为折射过程中宽度会变）
- rgb（继承光对象）
- pole（继承光对象）
- intensityset（对应路径中的不同光强）

上述所有末端的数据都是为了方便处理光线末端的折射反射过程。

对象的方法有：

- Draw（绘图）
- Addpath（添加路径）
- Reflect（输入法线，反射点，得到更新后的反射光线）
- Refract（输入法线，折射点，折射率，得到更新后的折射光线，如果发生全反射，则调用 reflect 函数）

绘图是根据储存的 widthset,intensityset 绘制各段光强和宽度。Reflect 主要是被平面镜调用，反射率由平面镜确定，而 refract 中包含了 snell 定律和菲涅尔定律，输入折射率后根据偏振选择折射率和反射率。这里在计算角度的时候，必须要对角度做一定的调制，因为其 2pi 的周期性，这里略过。

3，透镜对象

透镜对象主要是确定边界点的位置和折射率，所以实体属性较为简单：

- Type:lens
- N（折射率）
- Color
- Xlist（边界横坐标列）
- Ylist（边界纵坐标列）

除了控制类属性，对象方法为：

- Buildfacet, buildfacet2（构建一般边界）
- Buildcircle（构建圆边界）
- Draw
- Transfer
- Rotate
- Checkcross（输入子光线，得到最近的边界交点）
- Show_info

其中 buildfacet 是为了方便之后透镜的自由绘制。而圆边界是为了方便构造透镜而引入了的一个新函数。而 checkcross 函数是寻找子光和透镜边界相交的点的重要函数。

其计算的方法是：输入子光线，通过调用子光线的路径末端的点位置和路径末端的光线的方向，得到光线直线表达式，然后计算出和透镜的交点几何，去除不正确的方向的交点，然后寻找最近的交点，返回交点的值，法线方向，折射率。而光线可以处理这些信息，得到反射和折射后的光线。如右图 1 所示。

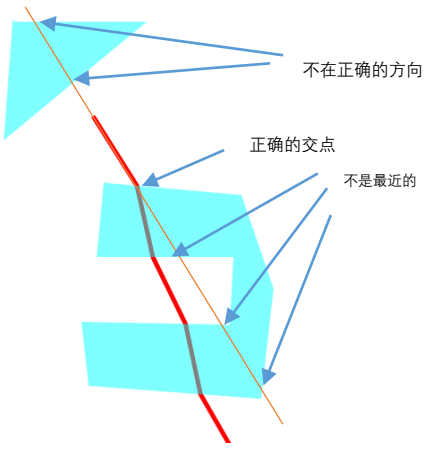

Figure 1

4，平面镜对象：

平面镜对象则更为简单，实体属性：

- Type

- Loc1,loc2
- Color
- R（反射率）

对象方法：

- Draw
- Transfer
- Rotate
- Checkcross
- Show_info

其中 checkcross 函数的功能和 lens 的一样，方便统一处理。

（2） 寻光部分

这一部分的主要功能就是根据现有的所有对象的属性或参数，计算出光路。也是主程序部分。实体对象通过三个列表储存：light_set,mirro_set,lens_set。操作逻辑是：

- 清除画布
- 初始化光线
- 光路计算，直到光路不再改变
- 绘制所有光学元件
- 绘制所有临时参考线
- 回调主函数

```
function main(){
    context.clearRect(0,0,canvas.width,canvas.height)
    init();
    while (light_on()){
        light_on();
    }
    draw_all(context);
    draw_tem(context);
    requestAnimationFrame(main);
}
main();
```

Figure 2

如右图 2 所示。其中重点的函数是 light_on()用来计算光路，每循环一次，子光列表更新一次，直到不能再更新。

Light_on()程序的逻辑是：

- 对所有光线
- 对所有子光线
- 所有元件对子光线做交叉检测 checkcross，返回最近的那个点。
- 光线折射或反射。
- 如果中途有反射或折射，则返回 changed=true，否则为 false，循环运行这个程序直到返回 false

（3） 交互部分

前面板的界面如图 3 所示：

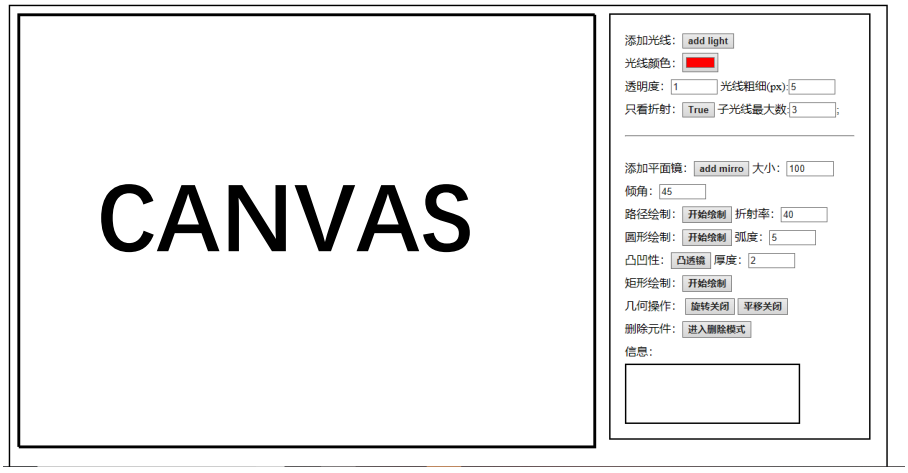

Figure 3

按钮事件有：

添加光线，添加镜子，开启折射反射模式，开启平移，开启旋转，开启任意路径绘制透镜，开启圆形路径绘制透镜，绘制凸透镜还是凹透镜，开启矩形绘制路径，开启删除模式

这些都是添加某种器件，或者是改变某种鼠标响应模式。所有的绘制，和平移操作都是用鼠标完成的，所以必须把不同的事件通过几个模式区分开来。

鼠标事件主要是两个函数，mouse_on_ele 判断鼠标和对象的相对位置，和 canvas.mouseonmove 作为鼠标在 canvas 上移动所诱发的事件。

在监测到 canvas.mouseonmove 事件后，根据 5 个模式对之后的事件作出不同的响应。

- 平移和旋转的几何操作
- 路径绘制透镜操作
- 圆形绘制透镜操作
- 矩形绘制透镜操作
- 删除操作

平移的操作逻辑：

当鼠标移动到某元件上时，鼠标改为 point。

- 当平移模式打开时，捕捉到 onmousedown，则改变对象的锚点 anchor 为当前点击的点，把对象设置为 transferable=true。
- 鼠标再次移动时，把新的位置和锚点的位置对比，得到平移矢量，通过对象的 transfer 函数对对象的坐标平移。并把 anchor 更新为当前点。
- 捕捉到 onmouseup，把对象设置为 transferable=false，不再平移

旋转的操作逻辑：

- 在旋转模式打开时，点击对象，则设置对象锚点 anchor 为当前点击对象，把对象的 rotatable=true，之后锚点不再改变，而在绘制过程中，锚点用一个小圆点显示出来，表示旋转操作的圆心。
- 之后移动鼠标后，再次点击对象之后，检测到 onmousedown，这时把当前点的值赋给 rotate_ref，这个点和平移的 anchor 功能类似，存储上一个鼠标所在的位置，当鼠标移动时，求这次移动的旋转角，调用对象的 rotate 方法，使得对象绕着 anchor 旋转，之后更新 rotate_ref。
- 当检测到 onmouseup，则把 rotate_ref 清空，暂停旋转。再次点击可以赋值 rotate_ref 重新开始旋转。
- 如何更改转动中心 anchor,可以点击 anchor 对应的小黑点,取消 anchor，之后便可以重新设置 anchor。

路径绘制的操作逻辑：

- 在路径模式开启的情况下，点击 canvas 上的任意一点，把值传递给 paintset，之后每点击一次，paintset 增加一个点。
- 在 paintset 的点数大于 1 时，主循环中的 drawtemp()开始起作用，绘制已经选择的 paintset 中的点和鼠标当前对应的点，鼠标对应的点传递给 locp。
- 当 paintset 点数大于 3，在点击距离起始点足够近的区域后，判定为封闭，则完成绘制，把对应的 paintset 传递给 lens_set，并初始化。

圆形绘制的操作逻辑：

- 在圆形绘制模式开启时，选定 canvas 上某点，传递给 Cpoint，作为圆

心，之后把鼠标当前值传递给 locp。

- 主程序中的 drawtemp()，根据所选的弧度，做出参考线，并且弧度参数传递给 angle_info
- 当鼠标再次点击后，通过 angle_info，和凹凸性，厚度，通过 lens 的 buildcircle，建立弧形的路径，可以得到凹透镜和凸透镜。
- 矩形绘制思路：
- 和其他类似，传递绘制参数 rect_p 作图。

删除模式的操作逻辑：点击某个对象，把这个对象从相应列表中删除。

三， 实例展示

上述对象部分储存在对应的对象.js 文件中，而主程序和交互部分储存在 light2D.js 中，而面板则是 light2D.html 程序中。

使用这个小程序，可以实现几何光学中的大部分现象，下面来举几个案例展示这个程序的功能：

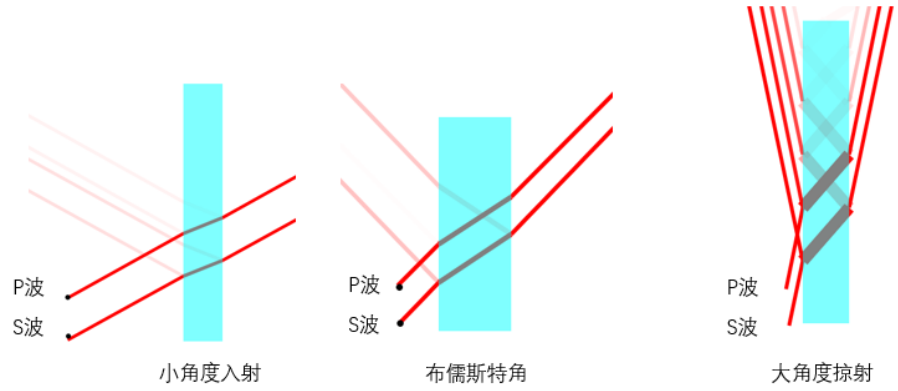

Figure 4

图四为不同偏振的光在界面反射，反射光和透射光强度随入射角的变化，可以为学生动态的展示反射光和透射光光强的变化。

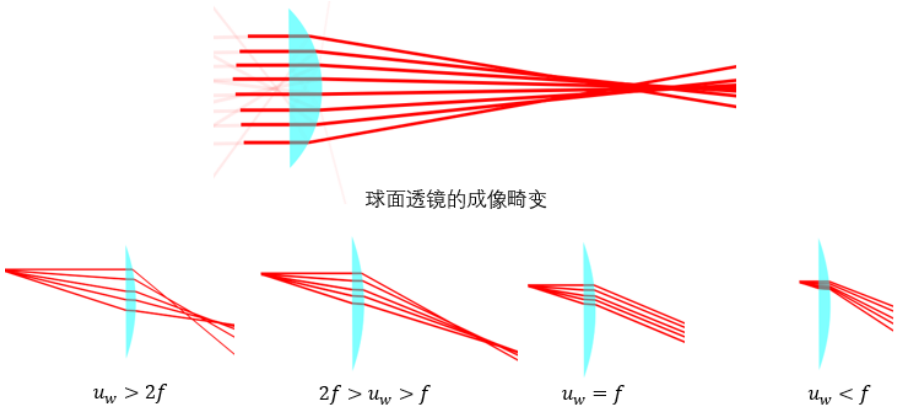

Figure 5

图五为透镜成像的展示，这些都是在每个界面上严格计算得到的透射光，可

以看到透镜成像过程中的像的畸变。

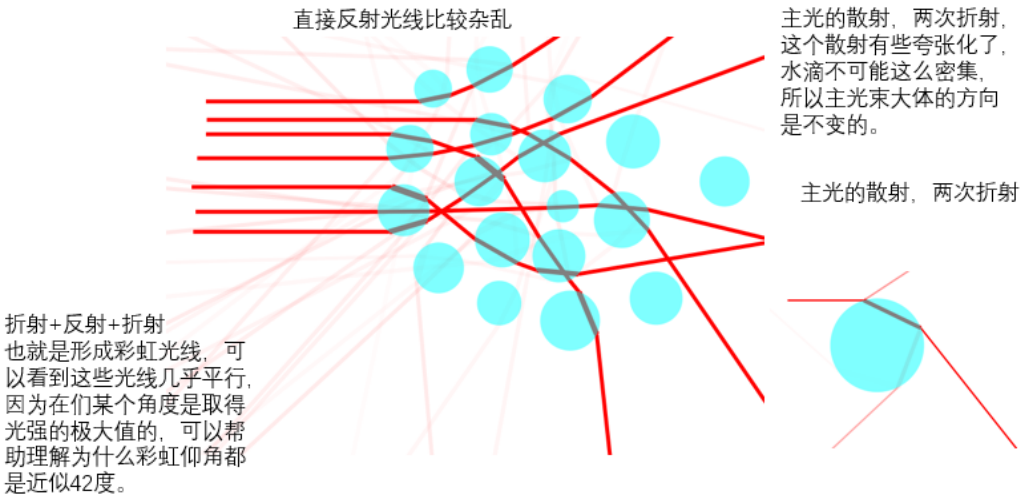

Figure 6

图 6 可以解释彩虹的仰角问题。由于这个软件可以方便处理多个元件和多数光纤，所以我们可以构建很多个小雨滴来模拟其对多光束的散射问题。

四， 改进方向

这个程序虽然说大体上已经成型，但是仍然有可以改进的地方：

1，光线在很粗的时候由于分段绘制，canvas 不能绘制多色线条使得图形很丑。改进手段：必须把绘图程序做非常大的改动，比如专门写一个绘制光线的程序，用三角形和矩形拼合出光线。

2，有必要加入集合体对象，就是可以把某些对象组合在一起，可以同时拖动，旋转。而这个集合体对象可以方便我们引入各种各样的光源类型，数据结构变为，一个系统有多个光源，一个光源有多条主光线，每条主光线又衍生多个子光线。这个光源可以是点光源，可以是平行光源，这样就可以方便处理一些成像问题，也方便我们处理多光线的排布问题。生成集合体的操作是通过绘制选择框，或者 control 选择来设置。这个又是一个大改动。

3，考虑是否要加入理想凸透镜，理想凹透镜对象，有时候我们不需要看相散等畸变。考虑把 mirror 对象改成容许曲面镜存在的镜子对象，mirror 对象设定过于简单。

4，无法实现波动光学。引入波动光学虽然只是要计算光程差，但是这个带来的直接难度是绘图的困难，如何绘制两束光的干涉图样。这个时候就必须在绘图的时候要计算每一个点的干涉后的光强，这个是很困难的，

canvas 绘图首先就必须要一个像素点一个像素点的绘制。所以从程序实现上，必须把光的路径储存的一个一维的列表改写成一个二维的形式才能更方便的编写，从而导致要引入波动光学必须从根本上改变程序的内容。几何光学可用的光学元件很少，没有波动光学丰富的内容。

5，绘图窗口大小的限制，可以采用比例尺的做法，实现整体的放大和缩小，而可以用 canvas 平移的做法，实现整体的移动，这样可以实现绘制窗口的无限大。

6，光路算法上的优化，在计算“彩虹”光路的时候，可以感受到卡顿，因为边界的点越多（比如圆形），计算量越大。可以再 checkcross 函数上做优化，比如使用一个矩形框标定元件的大概位置和大概大小，先粗略的判断时候有交点，然后再细化计算，可以节省很多运算量。

7，旋转和平移操作不能精确的控制，可以加入输入框，精确控制旋转的角度和平移的长度。还可以学习 solidworks 的配合操作，对垂直关系，平行关系，重合关系，对齐关系等对精确构建光学系统提供帮助。

8，加入 measure 工具箱，可以再 canvas 中起到一定的测量作用（和 solidwork 功能一样），比如实现焦距测量，物距测量。

9，辅助线的添加，除了绘图的辅助线，有时候光轴，反向延长线，发现等等都需要标定，对二维系统的构建和虚像的构建有帮助。

10，　……