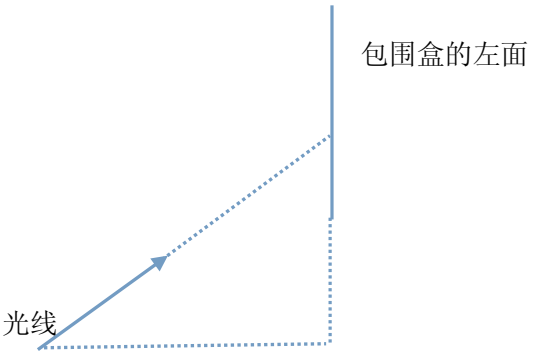
光线查询相关理论

1. 光线类 Engine::Ray
光线由两个要素决定，一个是光线的起始点，另一个是方向。

2. 光线和包围盒交叉的检测　Ray::intersects(const AxisAlignedBox& box)
第一步，检查光线的起点是否在包围盒的内部，若在则结束检测；
第二步，分别对包围盒的六个面进行判断：

1）如光线起点在包围盒左侧，且方向是向右的，则有可能交叉。具体检测方法如下：从光线向包围盒的左平面做延长线，延长线的长度可以有 x 方向来确定，然后求得延长线的终点 P。若该点 P 在做平面上则光线与左平面相交，可以返回。否则继续下面的判断。



2）如光线在包围盒的右侧，且方向是向左的，则可能和包围盒的右面交叉。检测方法类似上面。

3）其他四个平面的检测类似。

第三步，如果上面的两种检测（7 个）都没有检查通过，则表示该光线和包围盒是不相交的。

3. 如何由二维屏幕点击的位置确定一个世界坐标系的一个光线？
这个确定过程需要联系 OpenGL 中的坐标变换过程：我们绘制一个物体时制定的是一个局部坐标，局部坐标经过模型变换（场景节点的 transMatrix）变换到世界坐标系中去；世界坐标可以想象成是我们构建场景的一个虚拟的世界，这个虚拟的世界无边无尽，但是我们想要显示的却只是一块很小的区域，这个区域由我们设置的视椎体（view frustum）来确定，当我们给定一个世界坐标时，通过摄像机的模型视图矩阵变换就得到了一个 DC，即设备无关坐标（x,y,z 的范围都是从-1 到 1）。这个设备无关坐标具体如何变换成屏幕坐标则相对简单，一个简单乘法即可搞定。

现在我们就要按照相反的过程来进行：
首先，我们知道的是屏幕坐标，要变换成 DC 坐标，这个过称很简单，先将当前坐标规约到 0 和 1 之间，然后乘以 2 减去 1 即可（y 坐标的处理稍微有些不同，因为 OpenGL 中 y 坐标的方向与操作系统视窗中 y 的方向正好相反）；
其次，要将该 DC 变换成世界坐标，这个变换过程可以通过投影变换的逆矩阵得到：

$$a*T=d \rightarrow a=d*T^{-1}$$

所以我们可以从当前的摄像机获得模型视图矩阵，用当前的 DC 乘以该矩阵的逆矩阵即可得到世界坐标。

上面也介绍了，光线有两个要素来决定，一个是光线的起点，另一个是光线的方向。光线的起点的 DC 坐标就是 Z 值为-1，光线的方向我们只要再取光线上的一个点就好，这里我们取的是 Z 值为 0 的情况（如果取为 1 的话，在视景体的 far 为无穷的时候会算出无穷大的世界坐标，对光线的其他计算会造成影响）。这样，将这两个坐标变换到世界坐标系中，然后取起点和方向就变得较为容易。

4. 最后光线查询的内容就是用这样的一个光线和一个场景中的所有物体进行交叉检测，当然可以按照与光线起点的距离进行一定的排序然后返回给用户。当然，涉及到遍历场景，因此操作的代价还是很大的，因此要尽量减少其使用。