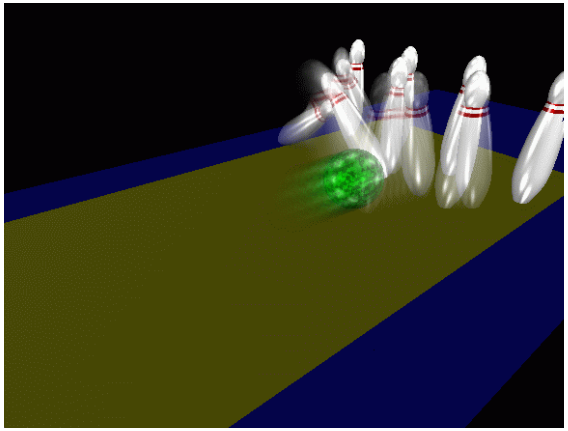
碰撞检测和处理
Collision Detection



Xiaogang Jin (金小刚)
State Key Lab of CAD&CG, Zhejiang University
Email: jin@cad.zju.edu.cn

「碰撞检测」Collision Detection, CD

- 碰撞检测是计算动画、计算机游戏和虚拟现实中最基本且非常重要的组成部分。
- 应用广泛，可用于
  - 计算机动画
  - 三维游戏
  - 基于物理的建模
  - 飞机和汽车驾驶模拟
  - 机器人
  - 路径和运动规划
  - 装配
  - 虚拟制造
  - **CAD/CAM**

「碰撞处理」

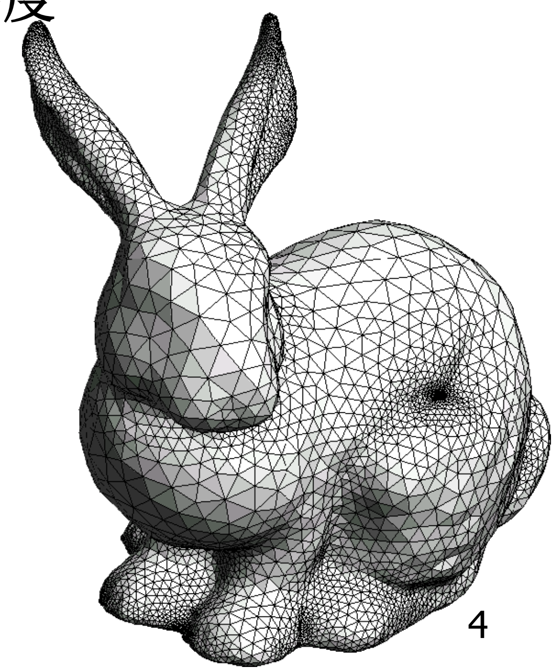
- 碰撞检测 (Collision Detection)
  - ✓ 返回两个或多个物体是否发生碰撞的布尔判断
  - ✓ 越快越好！

- 碰撞确定 (Collision Determination)
  - ✓ 找到物体之间的实际相交位置

- 碰撞响应 (Collision Response)
  - ✓ 针对两个物体之间的碰撞决定采取何种操作

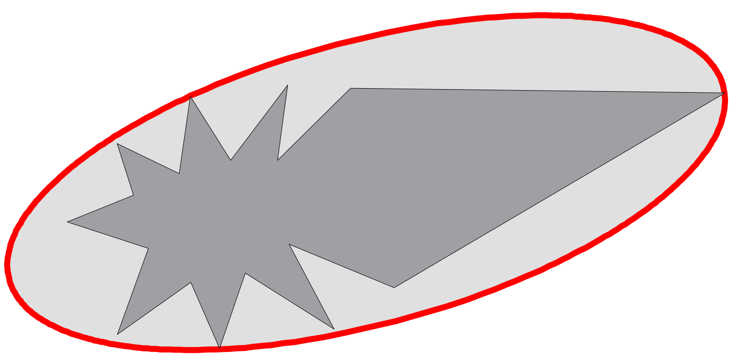
「碰撞检测」Collision Detection

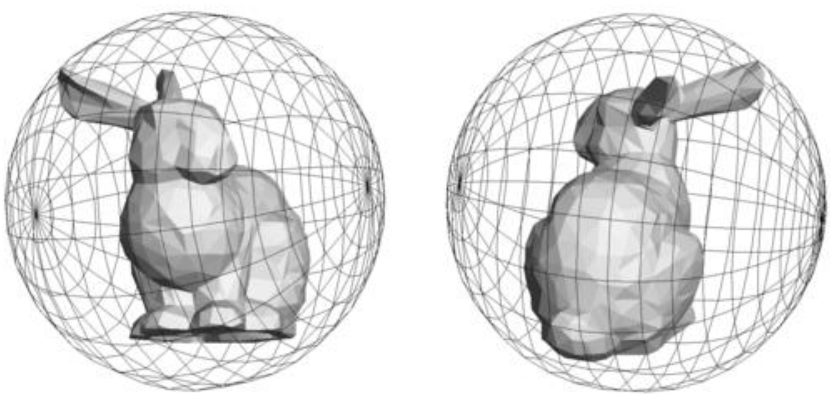
- 碰撞检测是非常复杂的，原因如下：
  - ✓ 碰撞物体的几何通常相当复杂，需要密集的计算测试
  - ✓ 对于**n**个物体，因为每个物体可能与任何其它物体碰撞，所以最直接的解决方法需要**O(n^2)**的时间复杂度
- 两个研究方向：
  - ✓ 复杂的物体必须简化
  - ✓ 减少物体测试对的数目



「简化的几何」

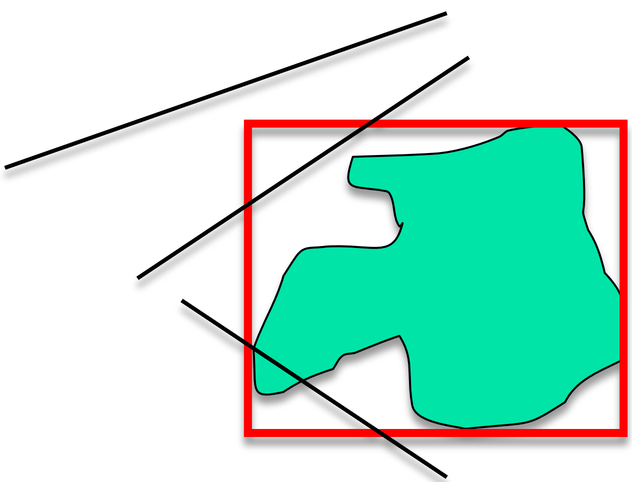
- 用简单的几何物体来逼近复杂的物体，如球体、长方体、椭球体等。
- 简单的形状测试的代价较低。

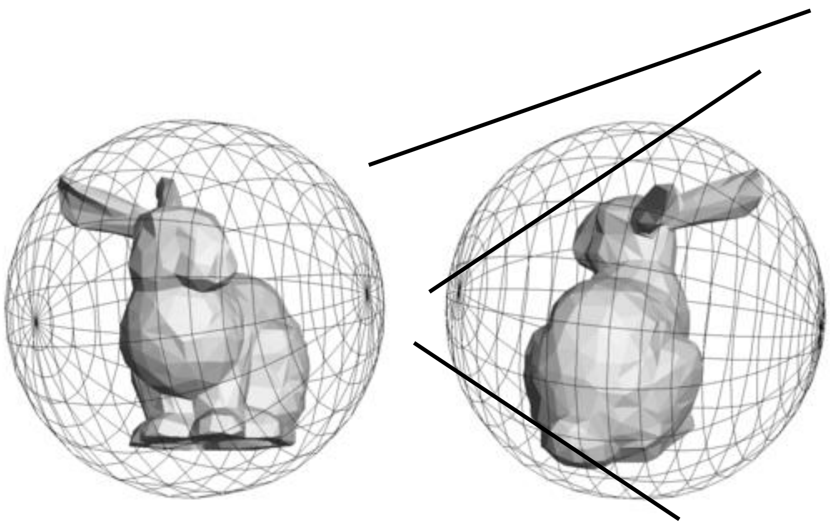




「包围体」BV: Bounding Volume

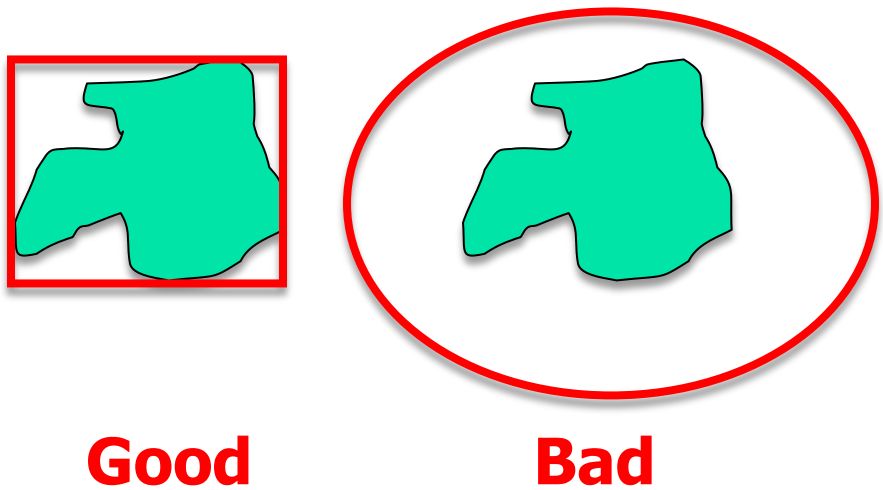
- 核心思路
  - ✓用一个**简单的**包围体把原物体包围起来
  - ✓如果一个物体与包围体不碰撞，则与包围体内部的物体也不碰撞
  - ✓通常，若要判断两个物体是否相交，首先测试它们的包围体





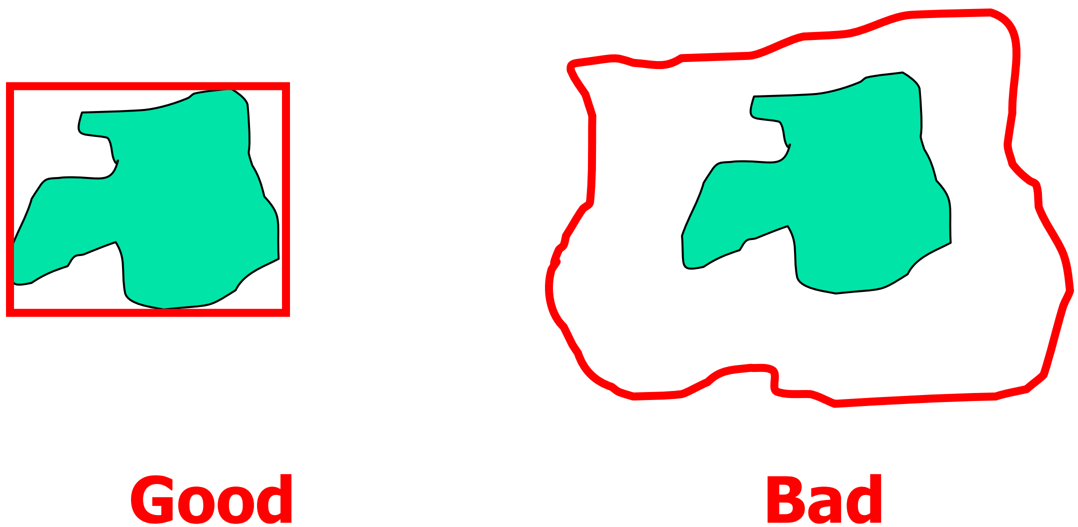
「如何选择包围体？」

- 有很多选择，每种选择都是某种折中
- 更好的包裹是更好的选择
  - ✓ 可以剔除一些“假”的相交



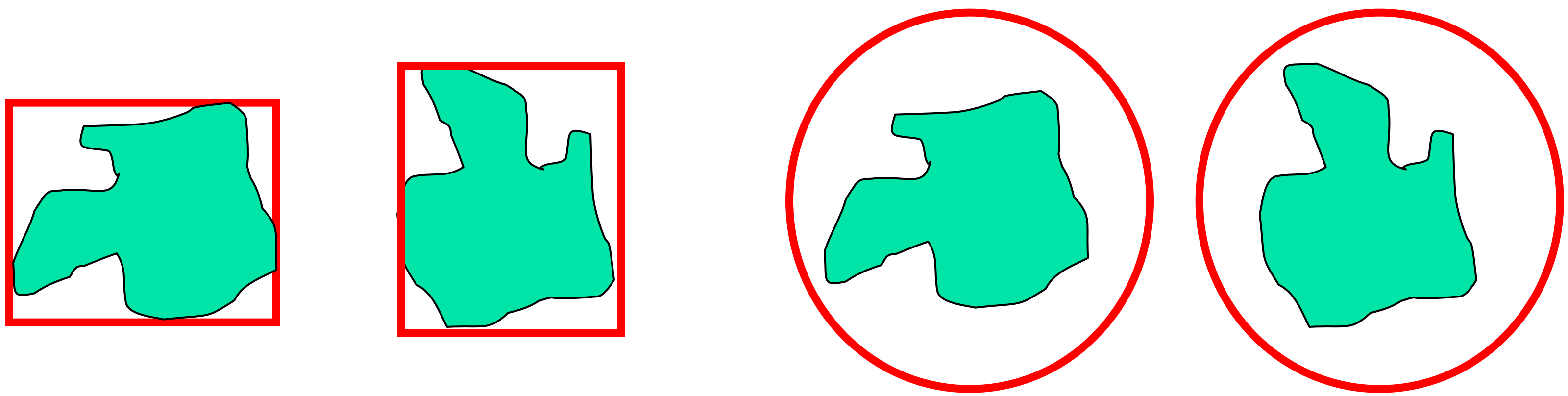
「如何选择包围体？」

- 有很多选择，每种选择都是某种折中
- 更好的包裹是更好的选择
- 物体的形状越简单，效果越好
  - ✓ 因为简单的形状容易计算



「如何选择包围体？」

- 有很多选择，每种选择都是某种折中
- 更好的包裹是更好的选择
- 物体的形状越简单，效果越好
- 具有旋转不变性的包围体更好

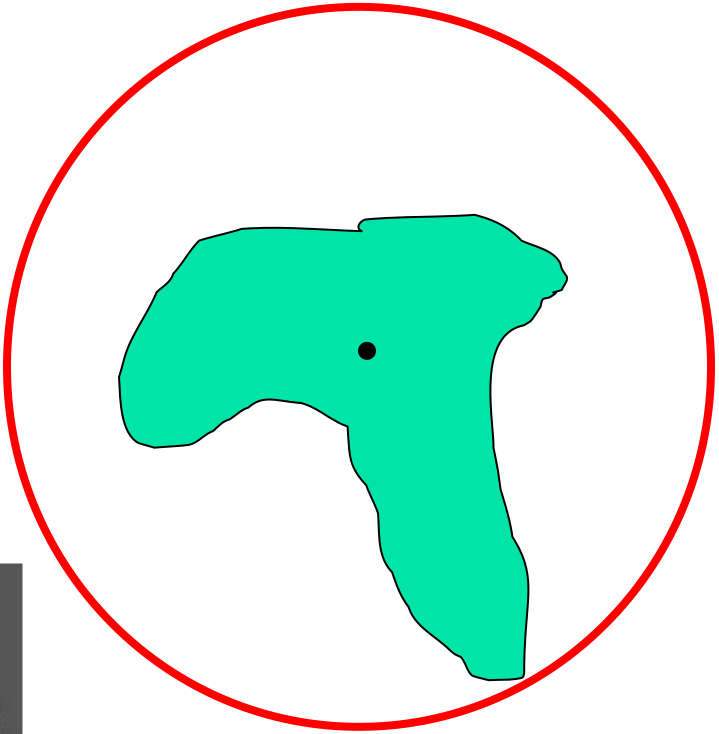


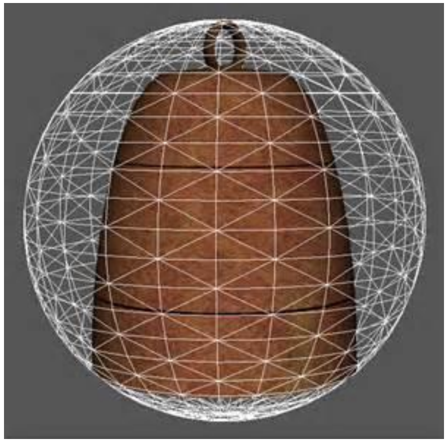
「碰撞检测算法中的图元」

- 碰撞检测算法通常建立在层次结构上，最后涉及判断两个图元是否发生碰撞。
- 图元类型：
  - 三角形
  - 球体
  - 轴对齐包围盒 (**Axis-Aligned Bounding Box，AABB**)
  - 有向包围盒(**Oriented Bounding Box, OBB**)
  - 离散有向多面体 (**Discrete Oriented Polytope，*k*-DOP**)
  - …

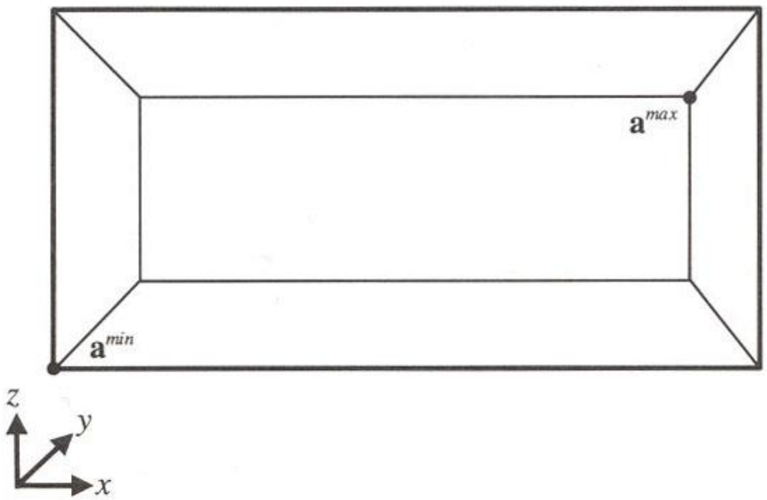
「常用包围体：球体」Sphere

- 具有旋转不变性
- 容易计算
- 用中心和半径来表示
- 但包裹得不紧密



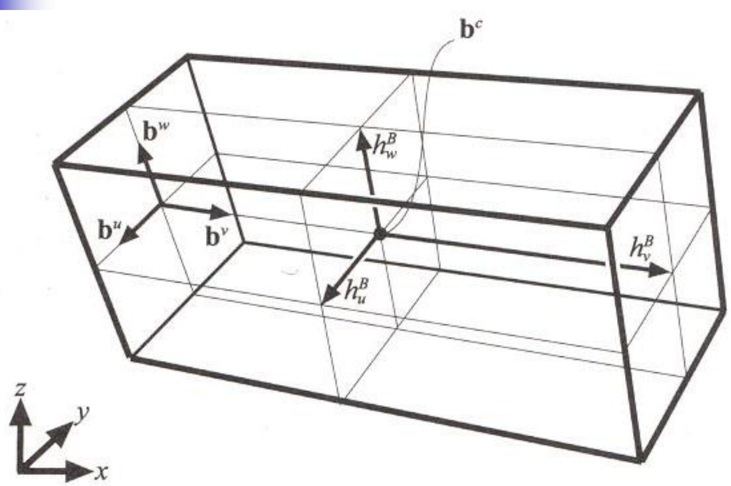


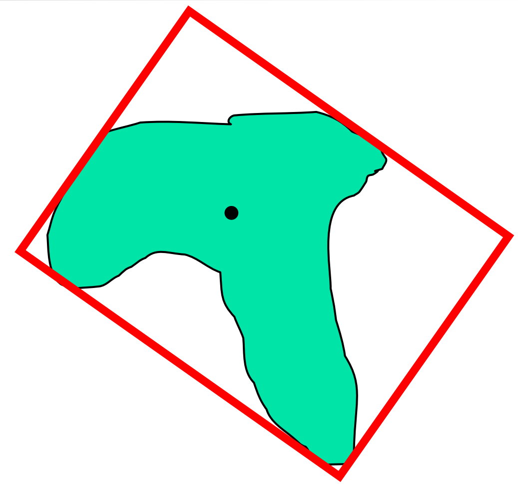
「AABB包围盒的定义」



- 轴对齐包围盒(**A**xis-**A**ligned **B**ounding **B**ox, **AABB**)
  - 是一个表面法向与坐标轴方向一致的长方体。
  - 可以用两个顶点$\mathbf{a}^{min}$和$\mathbf{a}^{max}$来表示物体A的AABB，其中$\mathbf{a}_i^{min}<\mathbf{a}_i^{max}$, $i \in \{x, y, z\}$
- 计算非常快、包裹性中等、物体旋转后必需更新包围盒(除非用一个包裹包围球的包围盒)。

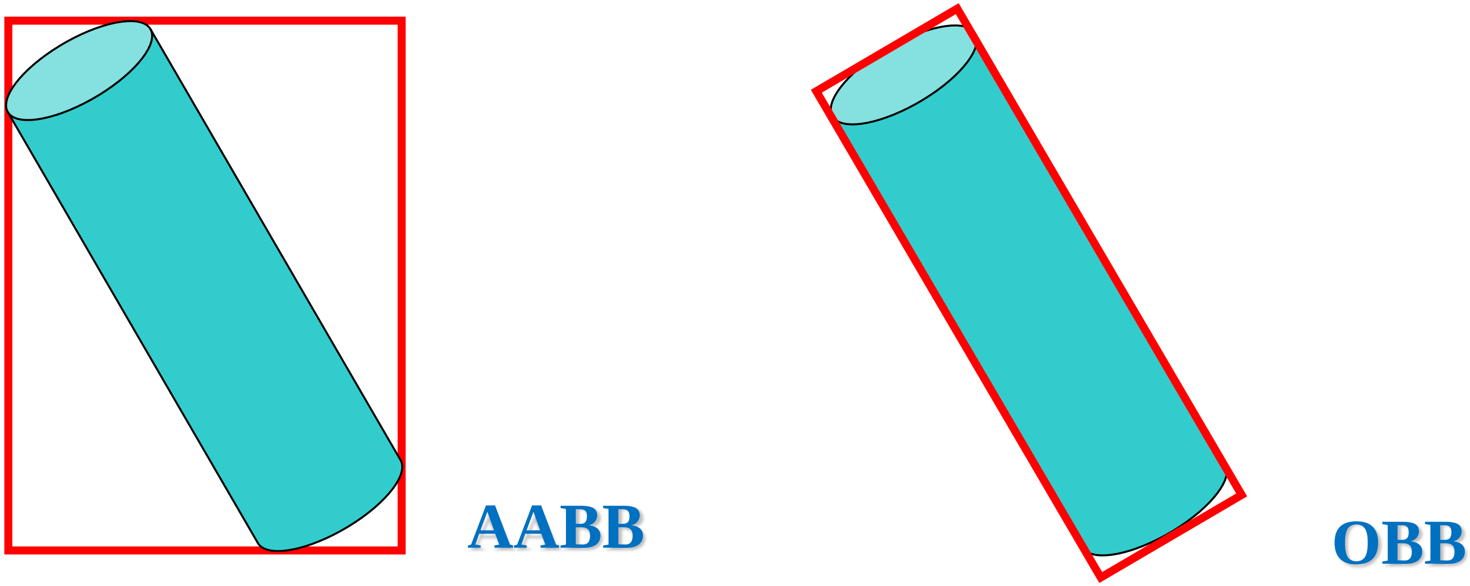
「OBB包围盒的定义」



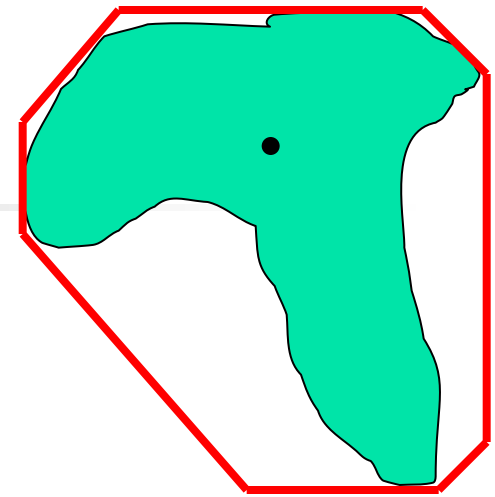


- 有向包围盒(**O**riented **B**ounding **B**ox, OBB)
  - 是一个表面法向两两垂直的长方体。是一个可以任意旋转的AABB
  - 可以用它的中心点$\mathbf{b}^c$、三个归一化向量$\mathbf{b}^u$、$\mathbf{b}^v$、$\mathbf{b}^w$以及半边长h_u^B、h_v^B、h_w^B来描述。
- 比AABB更紧密，但计算比AABB慢
- OBB随物体旋转，需更新三个归一化向量

AABB vs OBB



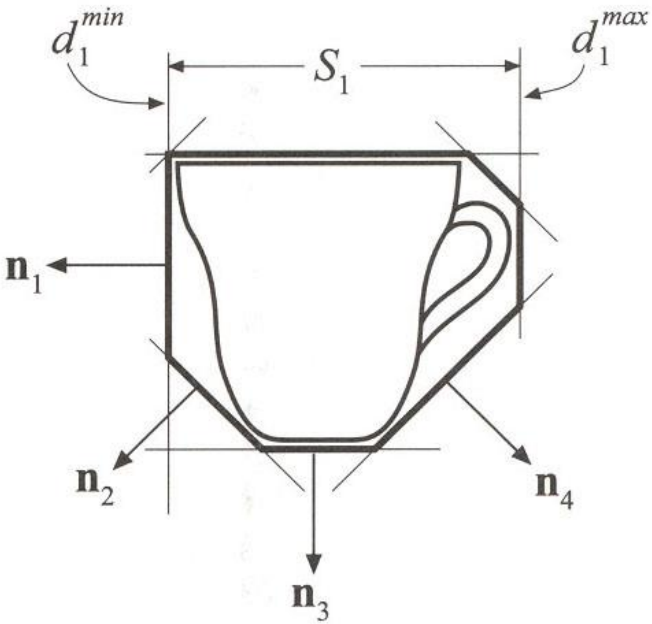
「离散有向多面体k-DOP」



- *K*-DOP：k-discrete oriented polytopes
  - Polyhedron：专指三维多面体
  - Polytope：任何维中由平面构成的图形

- 与AABB类似的思想, 但用更多的轴
- 包裹性比AABB 好，但计算量更大

「离散有向多面体k-DOP的定义」



一个茶杯的二维8-DOP示例。

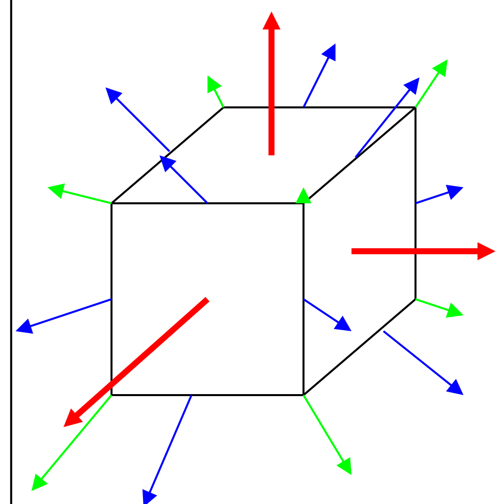
- 由$k/2$(k是偶数)个归一化法向$\mathbf{n}_i$来定义($1 \le i \le k/2$)，每个$\mathbf{n}_i$有两个相关标量值d_i^{min}和d_i^{max}，其中$d_i^{min} < d_i^{max}$。每个三元组($\mathbf{n}_i$, d_i^{min}, d_i^{max})描述一个平板层S_i，表示两个平面之间的空间。这两个平面是$\mathbf{n}_i \cdot \mathbf{x} + d_i^{min} = 0$和$\mathbf{n}_i \cdot \mathbf{x} + d_i^{max} = 0$。所有平板层的交集为***k*-DOP**的实际空间。

「k-dops轴的选择」

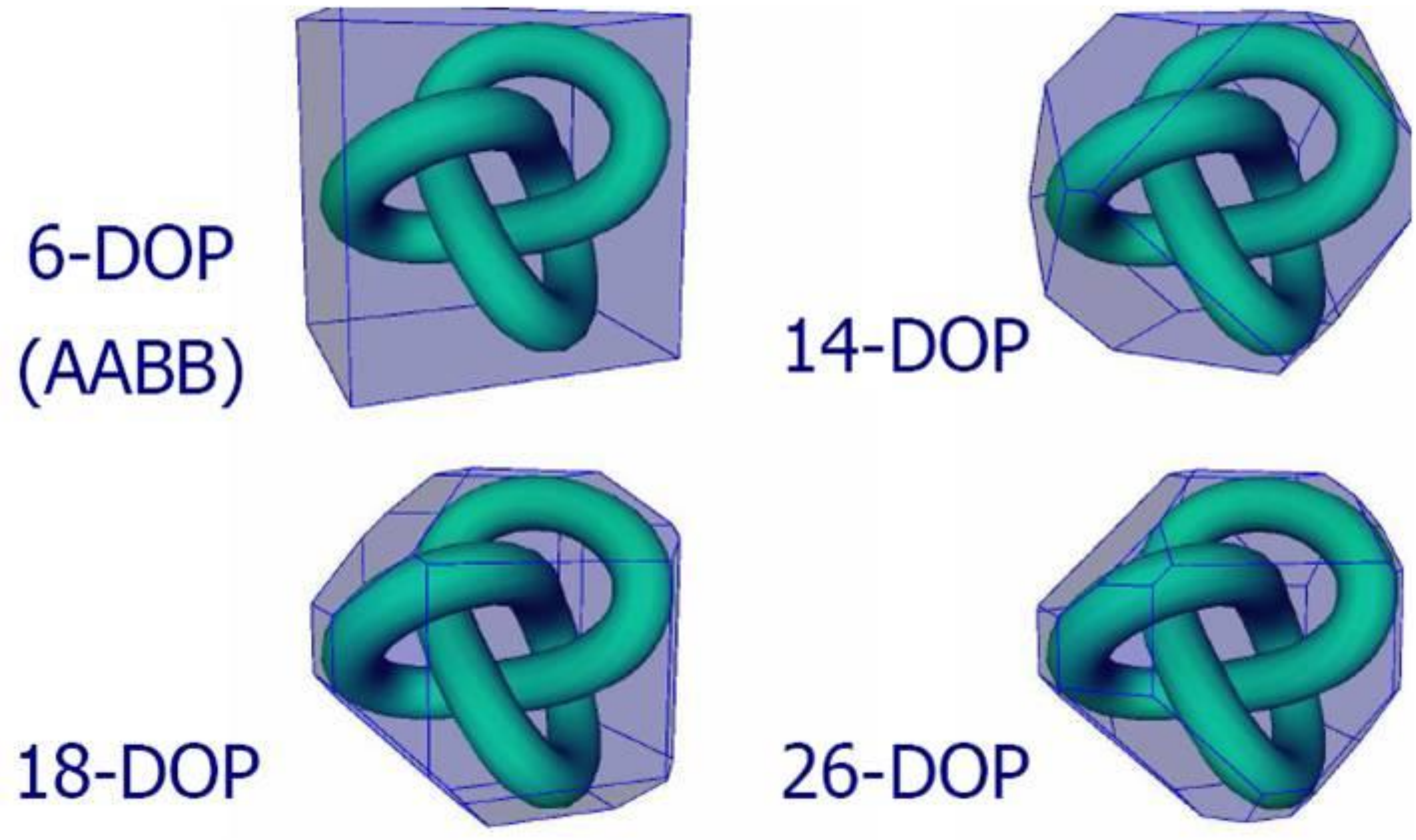
常用的轴: 从一个立方体中心出来的轴:

- Through faces: 6-dop
  - 与AABB一样
- Faces和vertices: 14-dop (6+8)
- Faces和edge centers: 18-dop (6+12)
- Faces, vertices, and edge centers; 26-dop (6+8+12)

- 比26-dop更大并没有多大帮助
- 实验结果表明， 14 或18-dop具有最好的性能

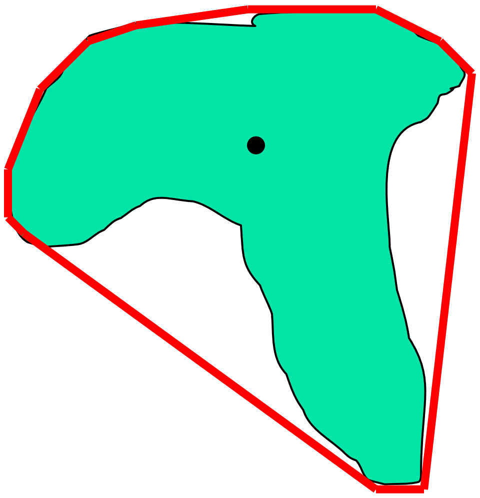


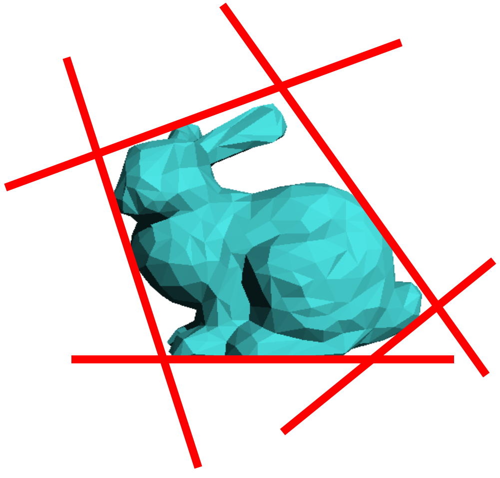
「k-dops中k的选择」



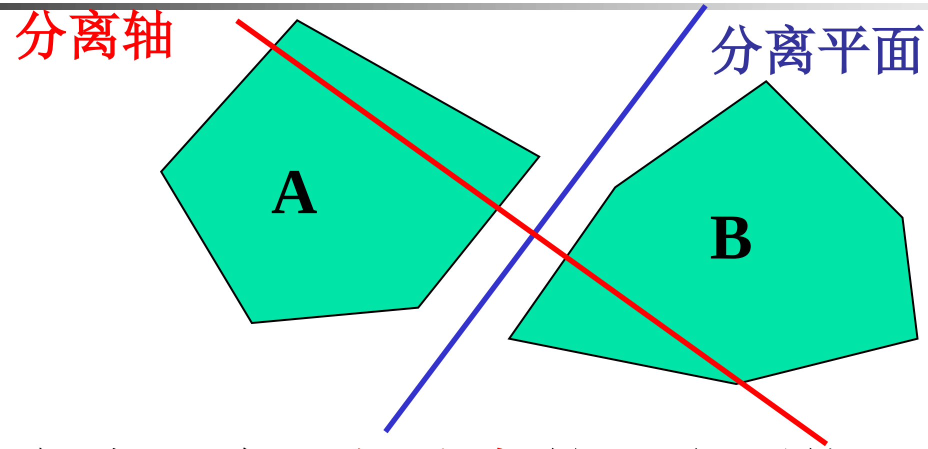
「常用包围体：凸包」Convex Hull

- 很好的紧密性 (最紧密的凸包围体)
- 存贮：形成包围盒的多边形
- 可以随物体旋转凸包
- 在有些应用非常有效





「分离轴定理」(Separating Axis Theorem, SAT)



- 对于任意两个互相分离的凸多面体A和B来说，存在一条分离轴，其中这两个多面体在这条轴上具有一定间隔而且在轴上的投影也是互相分开的。这条轴正交于下列其一(也就是由一个平面分开，该平面平行于下列其中一个平面)：
  - A的一个面
  - B的一个面
  - 多面体的一条边

注：这里指的广义上的凸多面体(包括线段和三角形等)

- 相交测试的优化方法
  - 排除检测：通过在早期进行一些简单的计算来判断射线或物体是否与其它物体完全分开
  - 投影：将三维物体投影到一个最优的正交平面上(xy, xz或yz)，然后在二维平面上进行处理。
- ε问题
  - 由于计算精度问题，相交测试常常会用到非常小的数字，用ε来表示。
  - 选择ε的值：具体问题具体分析。

「包围盒的创建」

- 给定一组物体，选择合适的包围盒非常重要，因为可以将相交测试的代价最小化。
- 任意一条射线与凸体相交的概率与该物体的表面积大小成正比。因为排除过程通常比直接计算射线与物体的交点要快，所以将表面积最小化可以有效提高相交算法的效率。
- 在碰撞检测算法中，应将包围体的体积最小化。

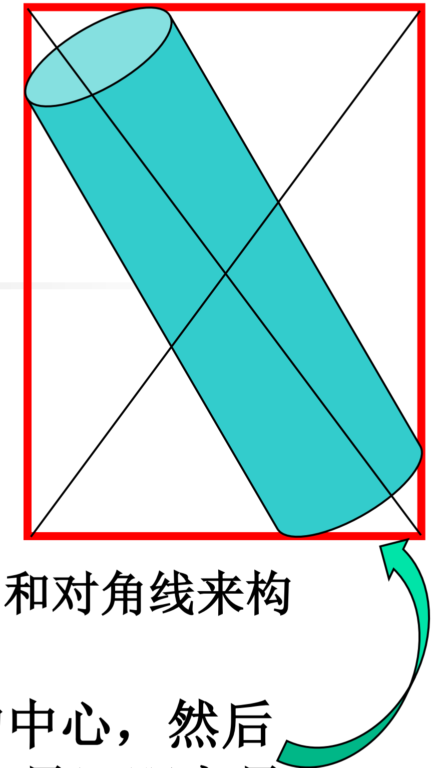
「AABB与k-DOP的创建」

- **AABB的创建**
  - 沿每条轴线方向分别取给定物体多边形组中顶点的最大值和最小值，就可得到**AABB**。

- **k-DOP的创建**
  - 将顶点投影到**k-DOP**的每条法线方向$\mathbf{n}_i$上，然后将投影的极小值和极大值分别保存在$d_i^{\min}$和$d_i^{\max}$中，这两个值定义了该方向上最紧密的平板层。所有这些值共同定义了一个最小的**k-DOP**。

「包围球体的创建」

AABB



- 算法很多，取决于速度和质量之间的折中。
- 恒时单遍算法(**Constant Time Single Pass**)
  - 先为多边形组构建一个**AABB**，然后用该**AABB**的中心和对角线来构造球体包围体。
- *改进*：从**AABB**的中心出发，将它作为球体包围盒的中心，然后再次遍历所有的顶点，找到距离最远的顶点，最后将最远距离最为球的新半径。
- **Ritter**的近似最优包围盒方法：先找到**x**、**y**、**z**轴线上最小和最大的顶点，然后从这三组顶点中找出距离最大的一组顶点，取这组的中心为球心，取它们之间距离的一半为半径。然后，遍历其它顶点，计算顶点到球心之间的距离**d**，如果这个顶点在球体半径**r**以外，则将球心朝着这个顶点方向移动**(d-r)/2**，同时将半径大小修正为**(d＋r)/2**。重复上述过程，直到把不在这个球体内的顶点和旧的球体全部包含在新的球体内。

「OBB的创建」

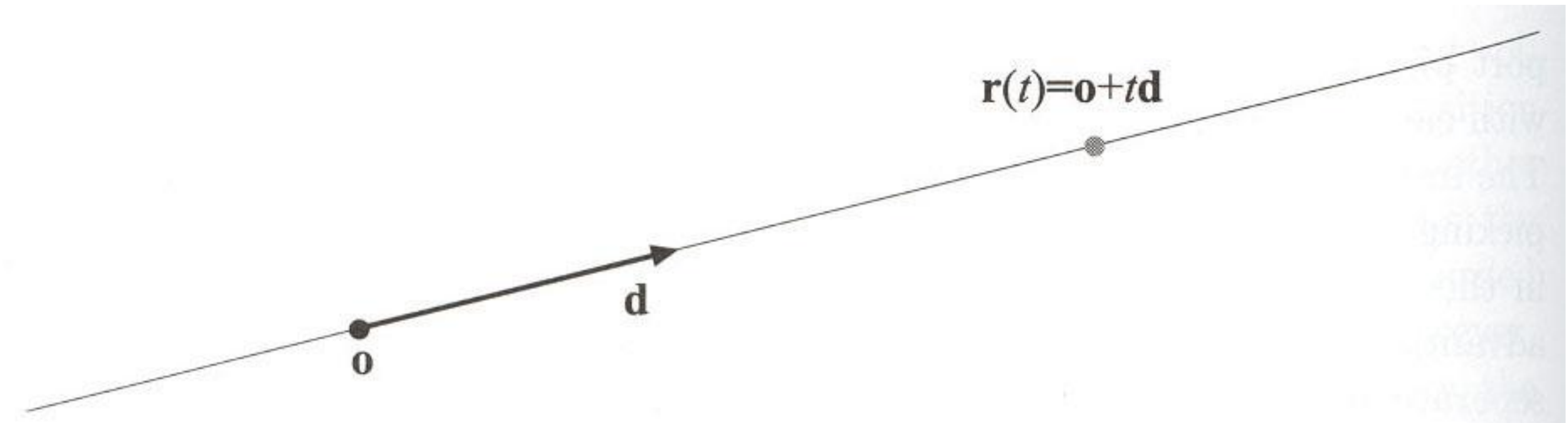
- **Gottschalk方法：**
  - 思想：首先从物体的凸包计算出一个方向，然后找到紧密贴合物体的OBB。
  - 凸包计算方法：QuickHull方法。计算复杂度为O(*n*log*n*)。
  - 方向计算方法：计算凸包的协方差矩阵，求其特征向量并将其归一化就是所球包围盒的方向向量。
  - OBB的中心和半边长计算方法：将凸包上的点投影到方向向量上，并求每个方向上的最大值和最小值。每个方向最大值和最小值的平均值决定了该方向的中心，最大值减去最小值的一半为该方向的半边长。

「求交测试的重要规则」

- 这些规则可以使得相交测试更快速、稳定、精确。
  - 尽可能利用简单的比较和计算来排除或确认各种相交类型，从而避免进一步的计算。
  - 尽可能充分利用上次的测试结果。
  - 如果使用了多种排除或确认测试，试着改变它们之间的内部顺序，这有可能产生一个更有效的测试结果。
  - 尽量搁置开销量大的计算，如三角函数、平方根、除法等。
  - 降低维数。（如把三维降为二维、二维降为一维）。
  - 尽量找出在相交测试之前可预先完成的计算。
  - 当相交测试比较复杂时，尽量先利用物体的包围球进行初步排除。
  - 上机实践进行计时比较，计时要采用真实的数据和测试环境。
  - 尽量使程序具有比较好的鲁棒性。能处理各种特殊情况，对浮点精度误差不敏感。

「定义和工具」

- 射线及其参数



一条射线及其参数：**r**(t)=**o**+t **d**，其中**o**为原点，**d**是射线方向(归一化为单位矢量)，t 为参数

「表面的定义」

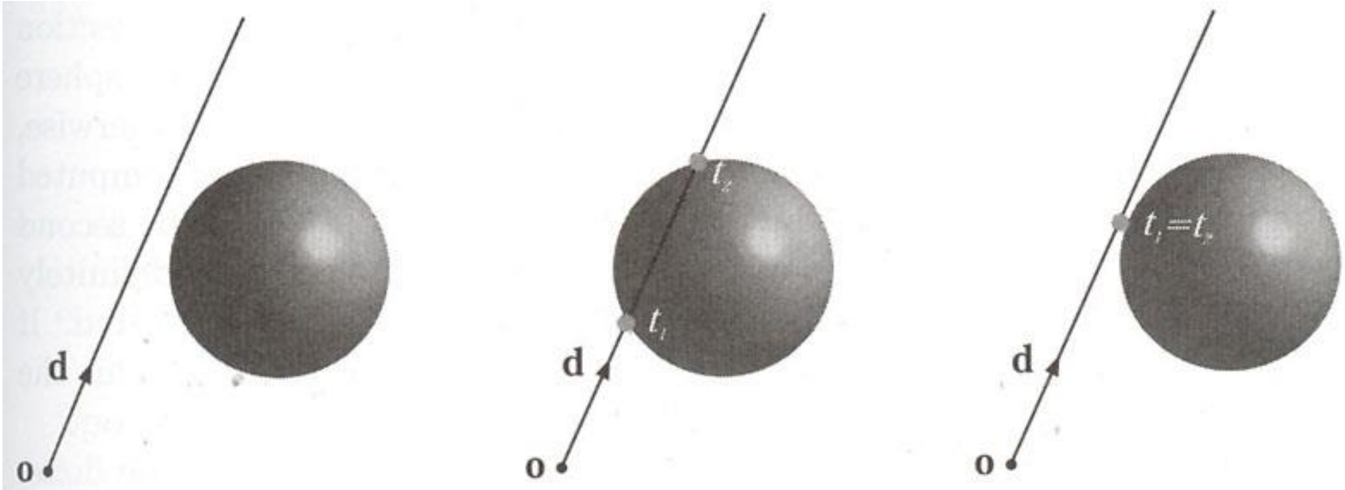
- 隐式表面 $f(\mathbf{P}) = f(p_x, p_y, p_z) = 0$
  - 例子: 中心在原点,半径为r 的球

$$p_x^2 + p_y^2 + p_z^2 - r^2 = 0$$

- 显式表面
  - 三角形▲$\mathbf{v}_0\mathbf{v}_1\mathbf{v}_2$表示为显示形式

$$\mathbf{t}(u,v) = (1-u-v)\mathbf{v}_0 + u\mathbf{v}_1 + v\mathbf{v}_2$$

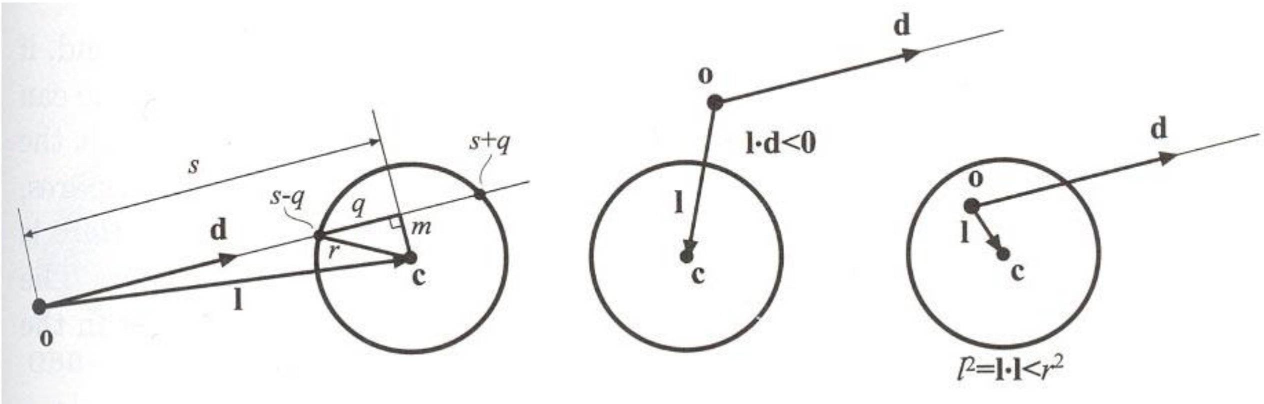
射线/球体相交测试



没有交点　　两个交点　　一个交点

- 数学方法
  - 把射线的参数方程代入球体的隐式方程，得到一个一元二次方程，利用方程的判别式进行无根判别和根的计算。

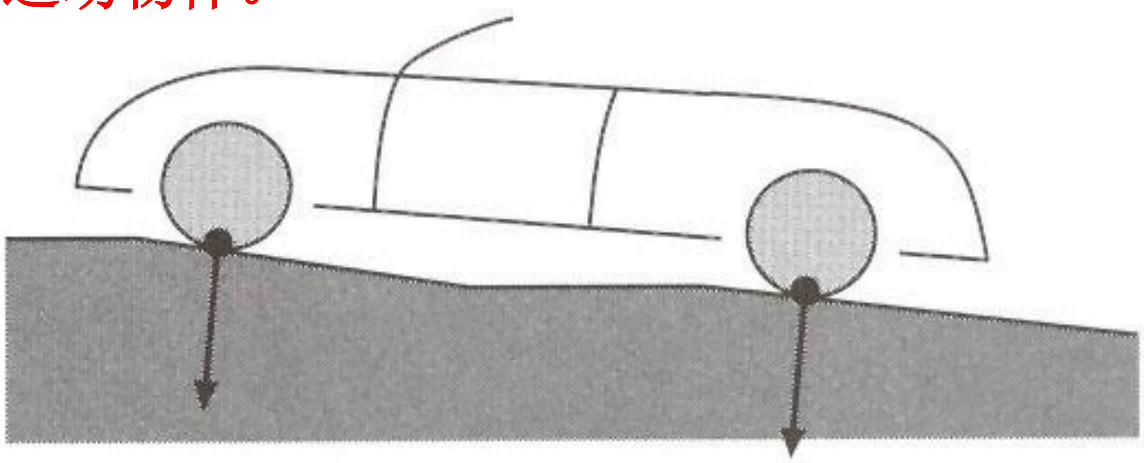
射线/球体相交测试



- 优化方法
  - 不需要对射线原点**o**后的相交进行计算。
    - 计算射线原点**o**到球体中心的向量**l**=**c**-**o**，其平方长度为l^2＝**l.l**。如果$l^2 < r^2$，则说明射线原点位于球体内部（见上图右），从而可以肯定这条射线肯定与球体相交。如果只需知道它们之间是否会相交，则结束，否则继续。
    - 计算**l**在射线方向的投影：s=**l.d**。如果s < **0**而且射线原点位于球体之外，则球体位于射线原点的后面，没有交点。(见上图中)
    - 利用勾股定理计算球心与投影点之间的平方距离：$m^2=l^2-s^2$。如果$m^2 > r^2$, 则射线与球体无交，否则肯定有交点。

使用射线进行碰撞检测

- 是一种特定环境下的快速碰撞检测技术，实际应用效果非常好。
- 例子：一辆小汽车沿着斜坡向上行驶，可利用路上的信息（路面形成的图元）来引导汽车向上行驶。但对于游戏等应用，并不需要这种复杂的碰撞检测。可使用一组射线来近似运动物体。



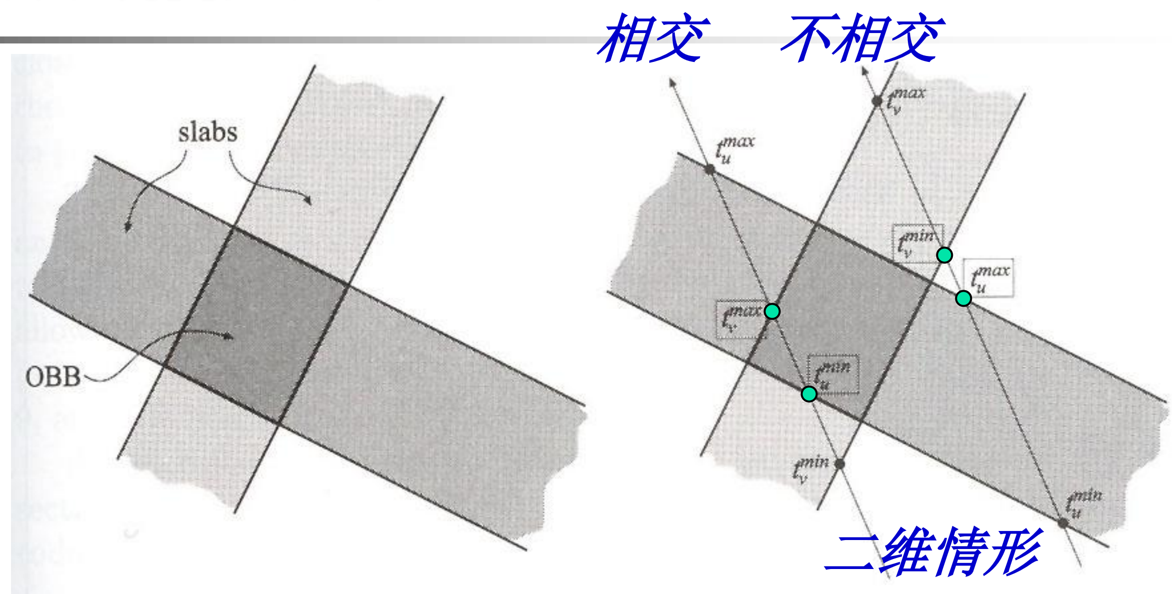
不是计算整辆测车子与环境（路面）之间的碰撞，而是在每个轮子上引出一条射线，然后对这些射线与环境进行相交测试。

使用射线进行碰撞检测

- 假设小汽车一开始在一个平面上，4个轮子是小汽车唯一与环境接触的地方。可在4个轮子上各生成一条射线，射线的起始点位于轮子和环境之间的接触点上，然后对轮子上的射线与环境进行相交测试。
- 如果从射线原点到环境的距离为0，则轮子在地面上；如果这个距离大于0，则轮子和环境没有接触；如果距离为负，则轮子陷在环境中。
- 可用这些值来计算碰撞响应：负距离可使小汽车（所在轮子处）向上运动，正距离可使小汽车向下运动。
- 为了避免射线在两个方向进行搜索，可将测试射线的原点向后移动直到它在环境包围盒的外面，然后再对这条射线和环境进行相交测试。

射线/平板层相交法(可以处理AABB、OBB、K-DOP等情况)

平板层(Slab)为两个相互平行的平面。



- 将长方体看成3个平板层。计算射线与平板层的交点。对于每个平板层，存在一个最小的t值和最大的t值，分别记做t_i^{min}和t_i^{max}，$i \in \{u,v,w\}$，计算

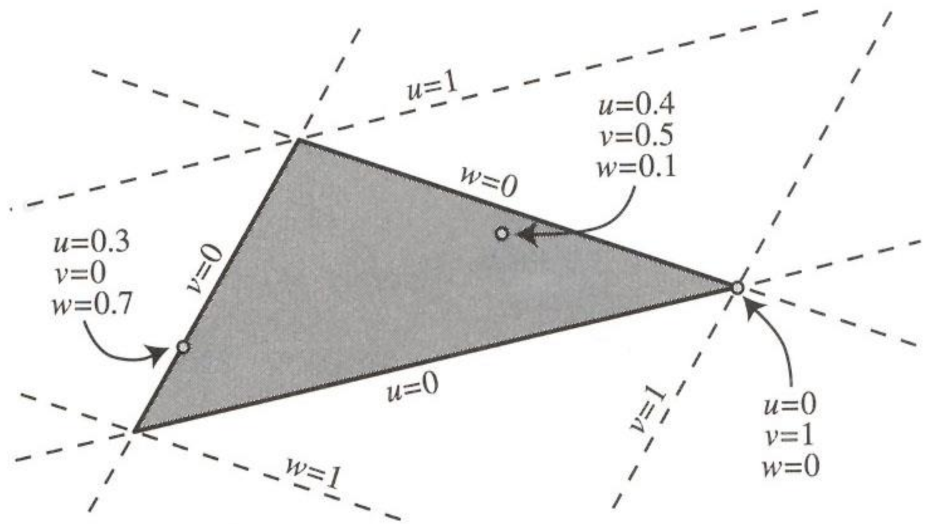
$$t_{\min} = \max(t_u^{min}, t_v^{min}, t_w^{min})$$

$$t_{\max} = \min(t_u^{max}, t_v^{max}, t_w^{max})$$

然后就可以进行巧妙测试：如果$t_i^{min} \leq t_i^{max}$，射线与长方体相交，否则不相交。

射线/三角形相交测试

- 原理：首先计算出射线与三角形所在平面的交点，然后将交点和三角形顶点投影到一个轴对齐平面（**xy, yz**或**xz**）上，并使得投影三角形面积最大化。
- 将三维问题变为二维问题，从而只需判断二维点是否在二维三角形内部。
- 三角形上的点$\mathbf{t}(u,v)$可以由下面的公式给出：
  $\mathbf{t}(u,v) = (1-u-v)\mathbf{v}_0 + u\mathbf{v}_1 + v\mathbf{v}_2$，其中$(u,v,w)$为重心坐标



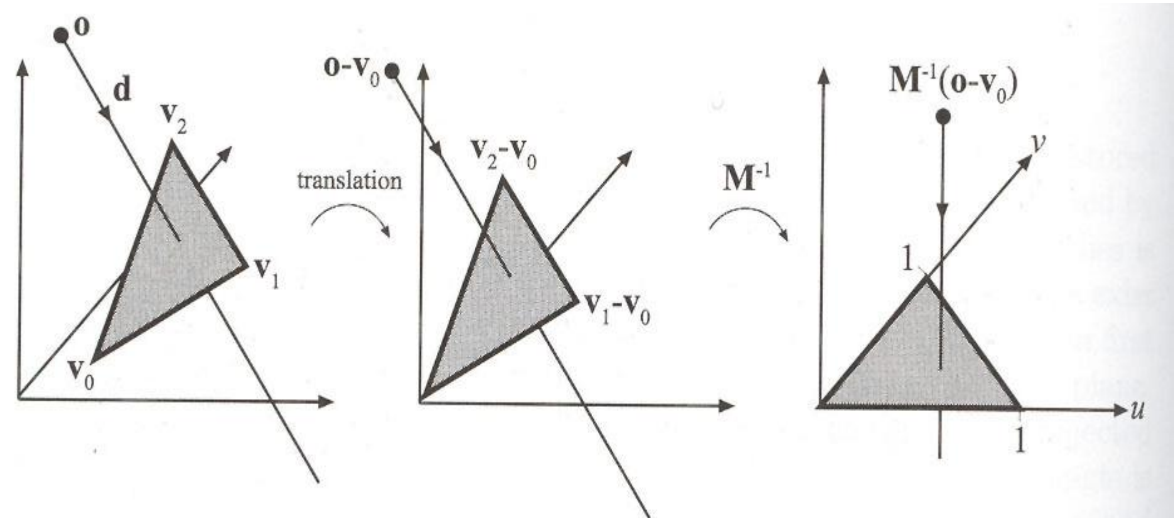
三角形的重心坐标

射线/三角形相交测试

- 射线$\mathbf{r}(t)$与三角形$\mathbf{t}(u,v)$之间的相交计算等同于对方程$\mathbf{r}(t) = \mathbf{t}(u,v)$的求解，得到线性方程组:

$$\begin{pmatrix} -\mathbf{d} & \mathbf{v}_1 - \mathbf{v}_0 & \mathbf{v}_2 - \mathbf{v}_0 \end{pmatrix} \begin{pmatrix} t \\ u \\ v \end{pmatrix} = \mathbf{o} - \mathbf{v}_0$$

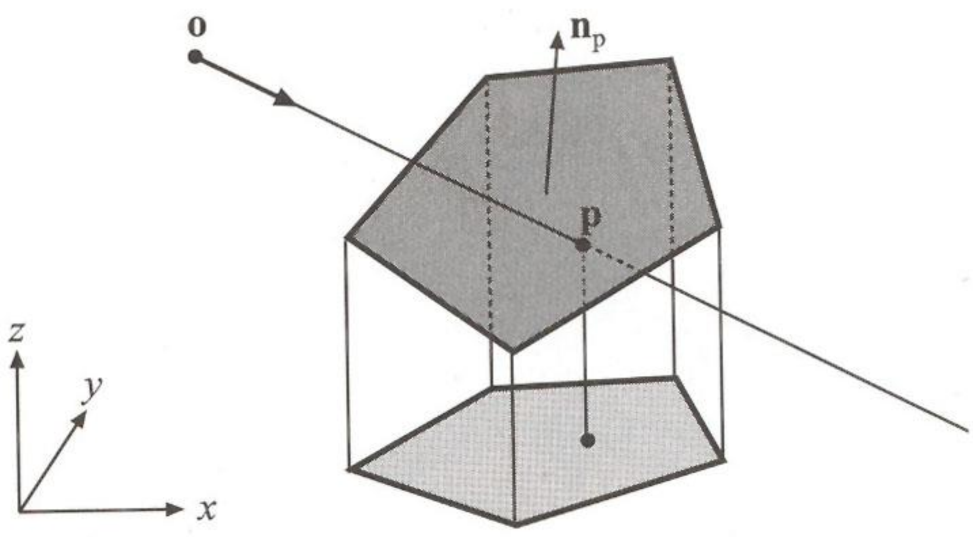
从而得到交点的重心坐标(u, v)和 t 值。



几何解释：把三角形平移到原点位置，然后在y方向和z方向将这个三角形变换为单位三角形，同时将射线方向与x轴方向对齐。

射线/多边形相交测试

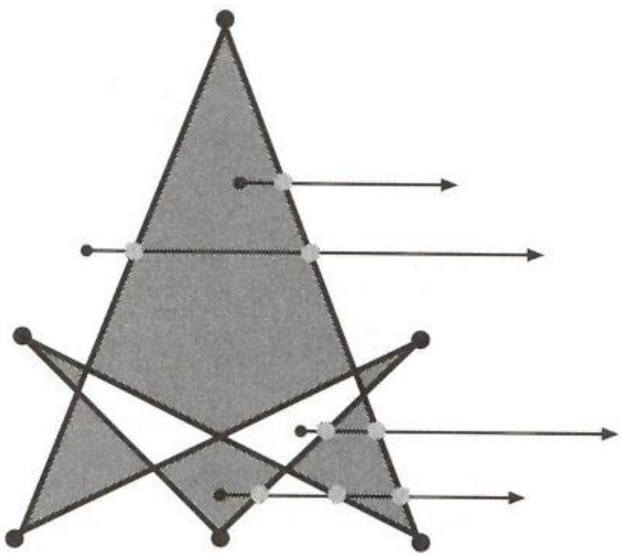
- 首先计算射线与多边形所在平面方程的交点
- 将交点和多边形的所有顶点投影到xy, yz或xz平面上（使得投影面积最大：利用平面的法向来判断，若x分量最大，则舍弃x分量，投影到yz平面），从而将三维降为二维。
- 在二维空间，判断交点是否包含在二维的多边形中。

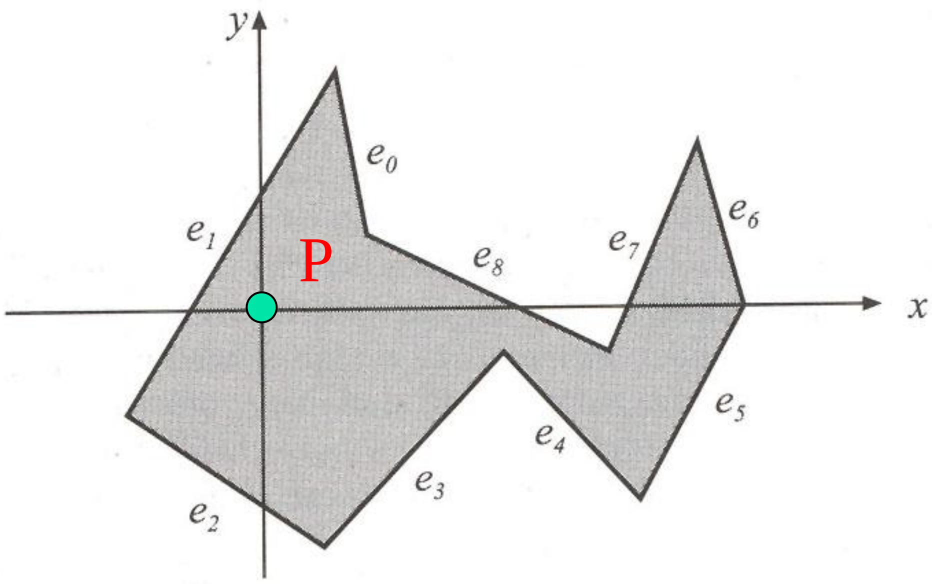


多边形顶点和交点P在xy平面上的正交投影（在xy平面上的投影面积最大）

点是否在二维多边形内部的 Crossing Test

- 基于Jordan 定理
  - 一条射线从某点出发，若与多边形的交点个数为奇数个，则点在多边形内部，否则在多边形的外部。





自交的凹多边形：*并非所有被多边形包围的部分就一定是多边形的内部。*

加速方法：将多边形平移-P, 这样通过计算与x轴正向的交叉点数量就可以确定P是否位于多边形内部。

平面/长方体相交测试

- 要确定长方体与平面∏: **n.x**+d=0之间是否相交，一种方法是将长方体的顶点代入该平面方程。如果所有的结果有正有负（或者为零），则表明这些顶点位于平面的两侧，因此存在相交关系。

- 针对**AABB**和**OBB**有更快的方法。

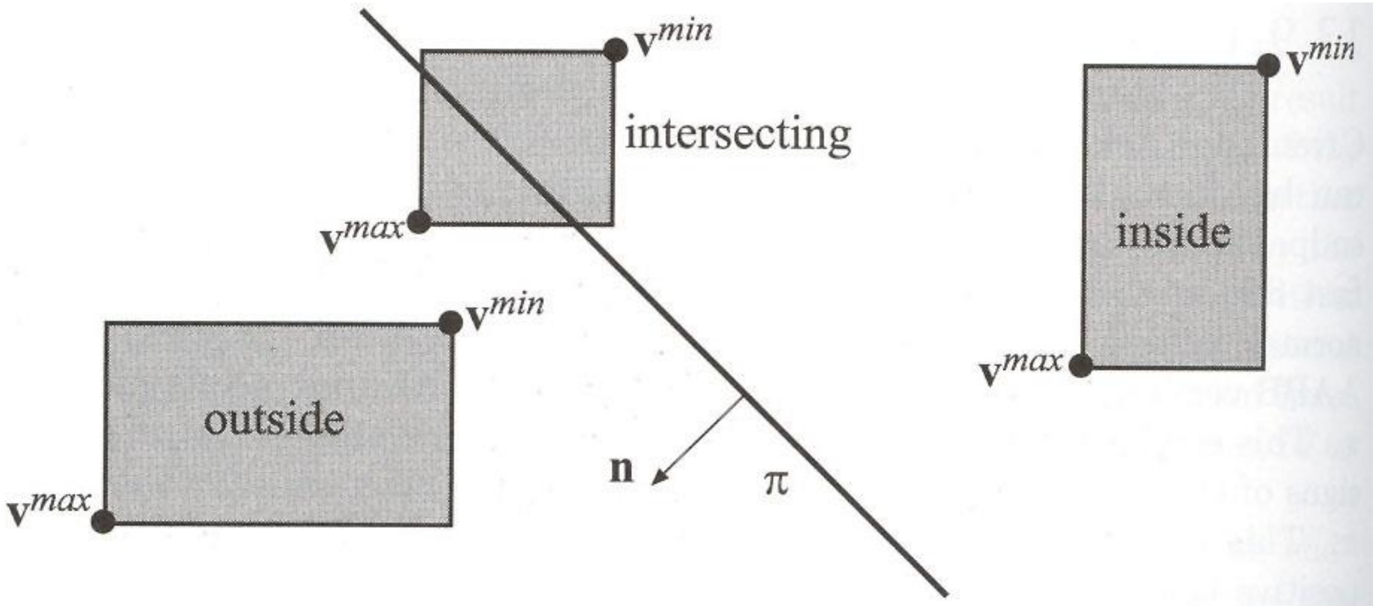
- 基本思想：只需要将两个顶点代入平面方程，这两个顶点构成了这个长方体的对角线。所取的对角线通过长方体的中心，而且*与平面的法向**n**对齐程度最好。*

平面/ AABB相交测试

- 给定一个由b^{min}和b^{max}定义的AABB，记做*B*，基于它可以生成4条不同的对角线，这些对角线都通过*B*的中心。
  - 首先找出哪条对角线与这个平面的法向最接近。
  - 然后将这条对角线的两个顶点代入平面方程中，从而可以测试出每个顶点位于平面的哪一侧。
  - 如果求解的结果符号相反，或者至少一个为0，则AABB与平面相交。

  - 注意: 如果AABB位于平面的正半空间或负半空间，则无交。如果v^{min}位于平面的正半空间，则v^{max}一定位于正半空间。

平面/ AABB相交测试



3个AABB对于给定平面的$\mathbf{v}^{\mathrm{min}}$和$\mathbf{v}^{\mathrm{max}}$，如果这对顶点位于平面的同一侧，则AABB与平面不相交，否则相交。注意：可先检测$\mathbf{v}^{\mathrm{min}}$，如果发现它与平面法向位于同一侧，则说明AABB“在外面”（即位于该平面的正半空间）。

"三角形/三角形"相交测试

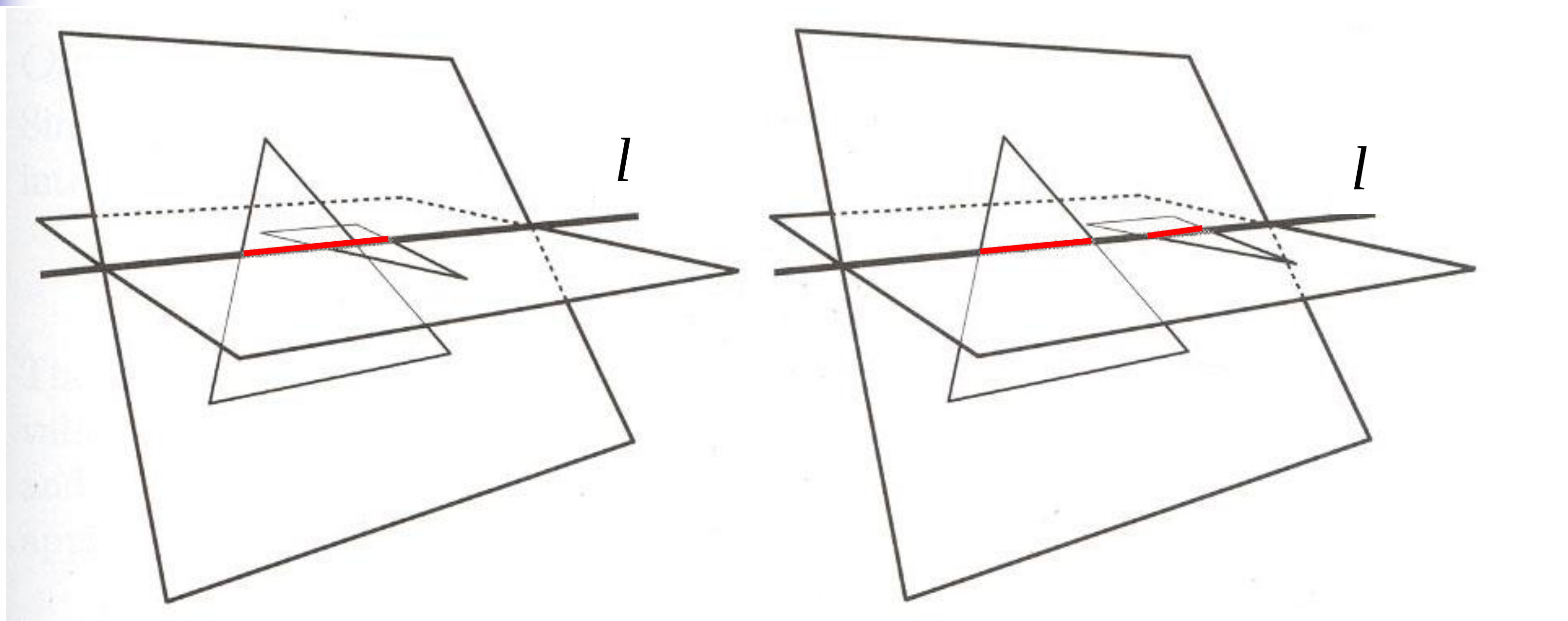
- 三角形是图形硬件最重要的绘制单元。在碰撞检测算法的最底层，一般会有一个程序来判断两个三角形是否相交。
- 通常仅关心它们之间是否会发生碰撞，而不需要具体的交点信息。
- 给定两个三角形$T_1=\triangle \mathbf{u}_0\mathbf{u}_1\mathbf{u}_2$和$T_2=\triangle \mathbf{v}_0\mathbf{v}_1\mathbf{v}_2$，它们分别位于平面$\prod_1$和$\prod_2$上，需要判断它们之间是否相交
  - 区间重叠法
  - ERIT方法

区间重叠法(Interval Overlap Method)

- 早期排除
  - 计算平面方程$\prod_2$，并把三角形T_1的顶点代入该平面方程，得到三角形T_1的顶点到平面$\prod_2$的符号距离
  - 若所有的距离不等于0，且这些值具有相同的符号(说明三角形T_1位于平面$\prod_2$的一侧），则不相交。
  - 对三角形T_2和平面方程$\prod_1$，进行同样的测试。

这两种早期排除测试可以节省很多计算量！！！

- 判断交线上是否含有两个三角形的公共部分
  - 如果所有的距离都等于零，则这两个三角形共面，需单独处理。（投影到二维空间进行处理）
  - 否则，平面$\prod_1$和$\prod_2$之间的相交部分应该是一条直线，可表示为$\mathbf{l}=\mathbf{o}+t\mathbf{d}$。
  - 可能发生的情形分两种，见下图



两个三角形及其各自所在的平面。

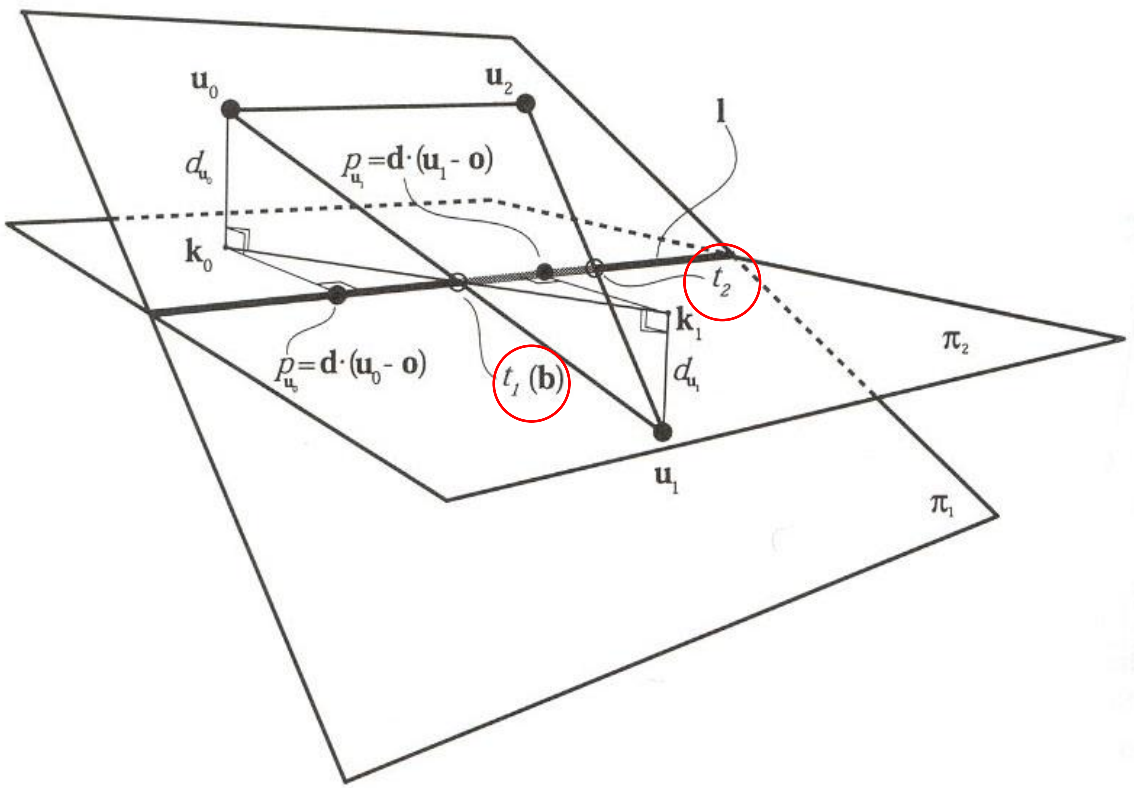
左图：与两个重叠三角形一样，沿直线l上的区间也相互重叠

右图：三角形之间不相交，直线上的区间也没有重叠

相交区间计算

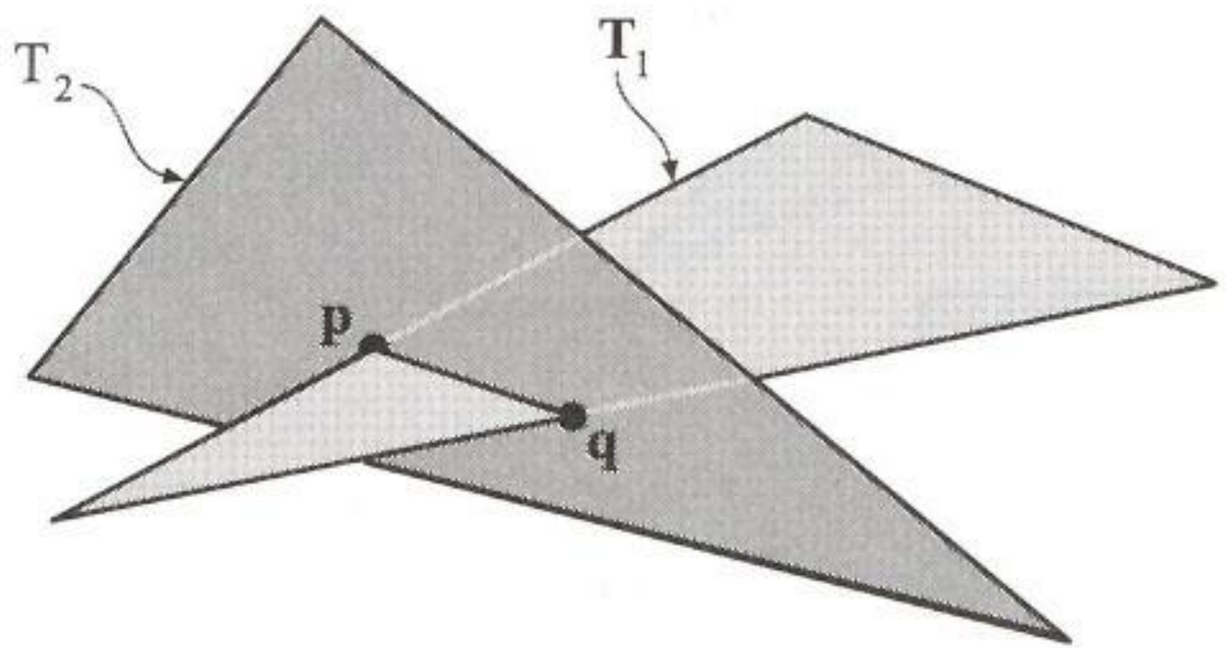
- 假设需要计算$\mathbf{T}_1$和$\mathbf{l}$之间的相交区间。
- 假设三角形的两个顶点$\mathbf{u}_0$和$\mathbf{u}_2$位于平面$\prod_2$的同侧，而另外一个点$\mathbf{u}_1$位于平面的另一侧。
- 利用几何关系可以算出直线$\mathbf{l}$与边$\mathbf{u}_0\mathbf{u}_1$之间的交点t_1，直线$\mathbf{l}$与边$\mathbf{u}_1\mathbf{u}_2$之间的交点t_2。$[t_1, t_2]$为三角形$\mathbf{T}_1$和$\mathbf{l}$之间的相交区间。
- 同理可以算出三角形$\mathbf{T}_2$和$\mathbf{l}$之间的相交区间。
- 如果区间相互重叠，则这两个三角形一定相交。否则不相交。

两个三角形之间的相交示意图



$\mathbf{u}_i$是三角形$\mathbf{T}_1$的顶点，$\prod_1$和$\prod_2$是三角形$\mathbf{T}_1$和$\mathbf{T}_2$所在的平面，$d_{\mathbf{u}i}$是从$\mathbf{u}_i$到平面$\prod_2$的有向距离，$\mathbf{k}_i$是$\mathbf{u}_i$在平面$\prod_2$上的投影点，$p_{\mathbf{u}i}$是$\mathbf{u}_i$在两个平面交线$\mathbf{l}$上的投影点。

ERIT方法



首先计算三角形T_1和T_2的两个交点p和q，如果p与q之间的线段完全包含在T2中，或者与T2的某条边相交，则这两个三角形相交，否则不相交。

算法描述

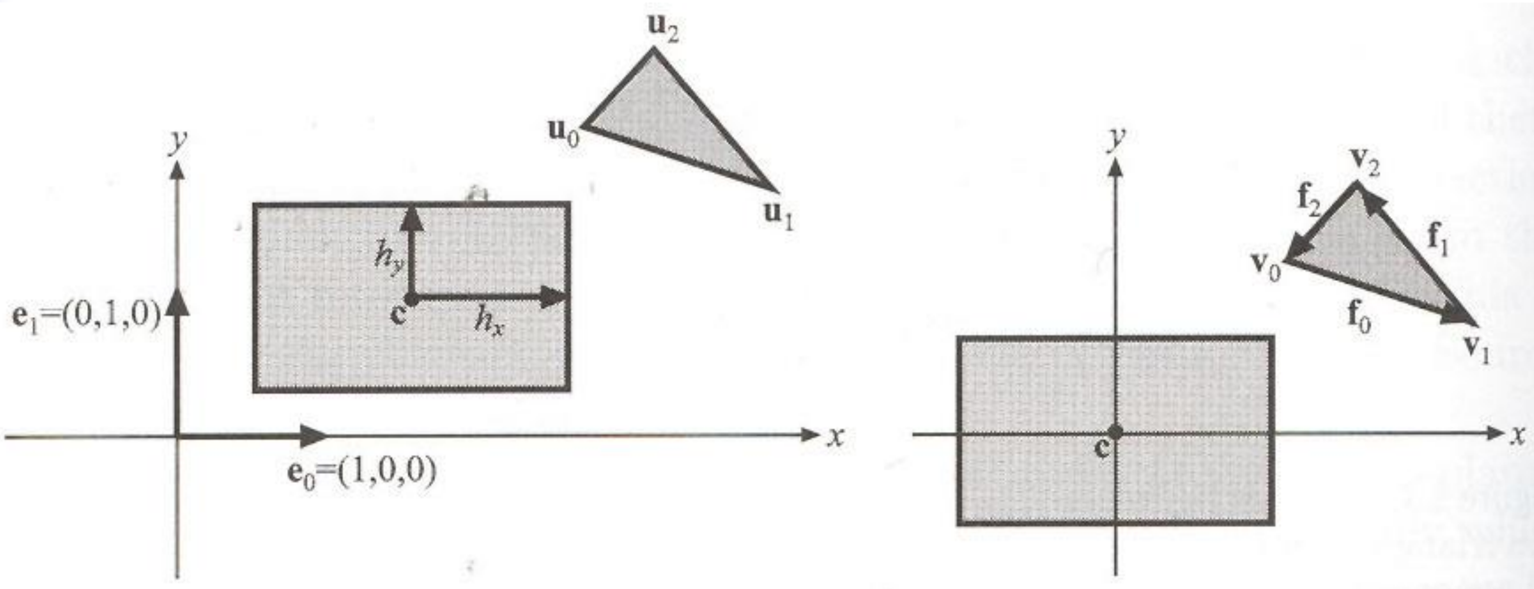
1. 计算出三角形T_2所在平面的方程
2. 如果三角形T_1上的所有点均在三角形T_2的同一侧，则不相交。
3. 如果两个三角形共面，则使用区间重叠法进行处理。
4. 计算出三角形T_1和平面$\prod_2$之间的交线，这条线位于平面$\prod_2$上。
5. 如果这条线段与三角形T_2相交，或者完全包含在T_2中，则T_1和T_2相交，否则不相交（可简单地计算**p, q**的面积坐标）。

“三角形/长方体”重叠测试

- 可用于
  - 体素空间的创建
  - 三角形与长方体的碰撞检测
  - 多边形与规范化视域体(Canonical View Volumes）的相交测试

- 我们将介绍Akenine-Moller提出的三角形/AABB的重叠测试方法（对OBB需要先变换）
- 原理：基于分离轴定理

“三角形/长方体”重叠测试中的符号表示



左图：为长方体和三角形在坐标系中原始位置

右图：对长方体和三角形进行了平移，使得长方体的中心与坐标系的原点重合

三角形/AABB的重叠测试

- 基于分离轴定理，对下面13条轴线进行测试
  - [3个测试] AABB的三个轴e_0=(1,0,0), e_1=(0,1,0), e_2=(0,0,1), 分别是AABB的3条法线
  - [1个测试] 三角线的法线n。只对与三角形法向最接近的长方体对角线的两个顶点进行测试
  - [9个测试] AABB的三个轴与三角形三条边的叉积

  - 在测试中，一旦找到分离轴，算法即可中止，返回没有相交信息。如果通过所有测试，则三角形与长方体重叠。

BV/BV相交测试

- 完全包含物体的封闭体称为包围体(**BV**)。**BV**的主要目的是为了提供更加简便的相交测试和更加有效的预先排除。

- 例如，如果要测试两辆汽车是否相撞，可先计算各自的**BV**，然后测试两个**BV**是否发生重叠。如果没有重叠，则可以确保这两辆汽车不会相撞(这是最常见的情况)，而不必对汽车上的每个图元进行测试，从而节省计算量。

- 最常见的包围体：球体、**AABB**、***k*-DOP**、**OBB**

球体/球体相交测试

- 计算出两个球体球心距离
- 如果这个距离大于两个球体的半径之和，则不相交；否则，相交。

球体/长方体相交测试：

- 主要思想：在AABB上找到距离球心最近的一个点。用球心对AABB的每条轴进行一次一维测试。
- 对每条轴，用球心对AABB的该轴边界进行测试。如果在球心AABB的边界之外，则计算球心与AABB在这条轴的距离的平方。
- 在三条轴线全部计算完之后，将这些距离的平方和相加，并与球体半径的平方进行比较。
- 如果平方距离和小于球体半径平方，则最近点位于球体内部（相交）；否则不相交。
- 将球心坐标变换到OBB的轴上去，即可处理OBB

球体/长方体相交测试伪代码（带加速）：

bool **SphereAABB_intersect**($\mathbf{c}, r, A$)
returns({OVERLAP, DISJOINT});

$d = 0$
for each $i \in \{x, y, z\}$
    if $((e = c_i - a_i^{\min}) < 0)$
        if $(e < -r)$return (DISJOINT);
        $d = d + e^2$;
    else if $((e = c_i - a_i^{\max}) > 0)$
        if $(e > r)$return (DISJOINT);
        $d = d + e^2$;
if $(d > r^2)$ return (DISJOINT);
return (OVERLAP);

AABB/AABB相交测试

- 因AABB与轴线对齐，所以只需要两个点就可以描述这个体。由于AABB非常简单，因此通常将它应用于碰撞检测算法中。
- 思想：每个轴上区间是否相交

Bool AABB_intersect(A,B)

returns({OVERLAP, DISJOINT}) ;

for each i $\in$ {x,y,z}

if($a_i^{\min} > b_i^{\max}$ or $b_i^{\min} > a_i^{\max}$ **)**

return(DISJOINT) ;

return(OVERLAP) ;

k-DOP/k-DOP相交测试

- 一个***k***-**DOP**是一个凸多面体，各个面由***k***个固定的法向确定，其中法向以外的半空间不作为包围体一部分来考虑。
- **AABB**实际上是一个特殊的**6-DOP**。当***k***值增大时，包围体越来越接近凸包，从而更接近物体。
- **Klosowski**等指出，对于适当的***k***值，两个***k***-**DOP**相交测试速度要比两个**OBB**的相交测试速度快一个数量级。
- 假设两个***k***-**DOP**分别为**A**和**B**，需要对它们进行相交测试，这时需对每一对平板层（S_i^A, S_i^B）进行重叠测试，而$S_i = S_i^A$) S_i^B是一个非常容易解决的一维区间重叠测试。
- 如果在任何情况下S_i为空集，则这两对平板层相互分离，测试过程结束。如果所有S_i均不为空集，则可以确定这两个***k***-**DOP**相交。

k-DOP/k-DOP相交测试算法描述

- KDOP_Intersect($d_1^{A,min}$, ...,$d_{k/2}^{A,min}$,$d_1^{A,max}$, ... , $d_{k/2}^{A,max}$,

  $d_1^{B,min}$, ..., $d_{k/2}^{B,min}$,$d_1^{B,max}$, ... , $d_{k/2}^{B,max}$)

  returns ({OVERLAP, DISJOINT});

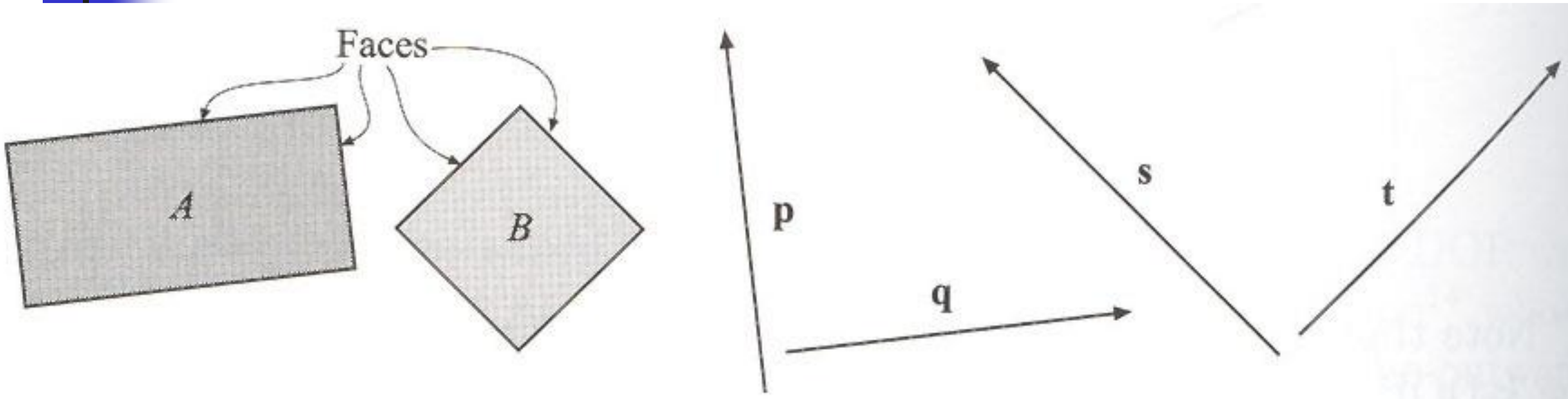
  for each i {1,2, ... ,k/2}

  if ($d_i^{B,min} > d_i^{A,max}$ or $d_i^{A,min} > d_i^{B,max}$)

  return (DISJOINT);

  return (OVERLAP) ;

“OBB/OBB”相交测试

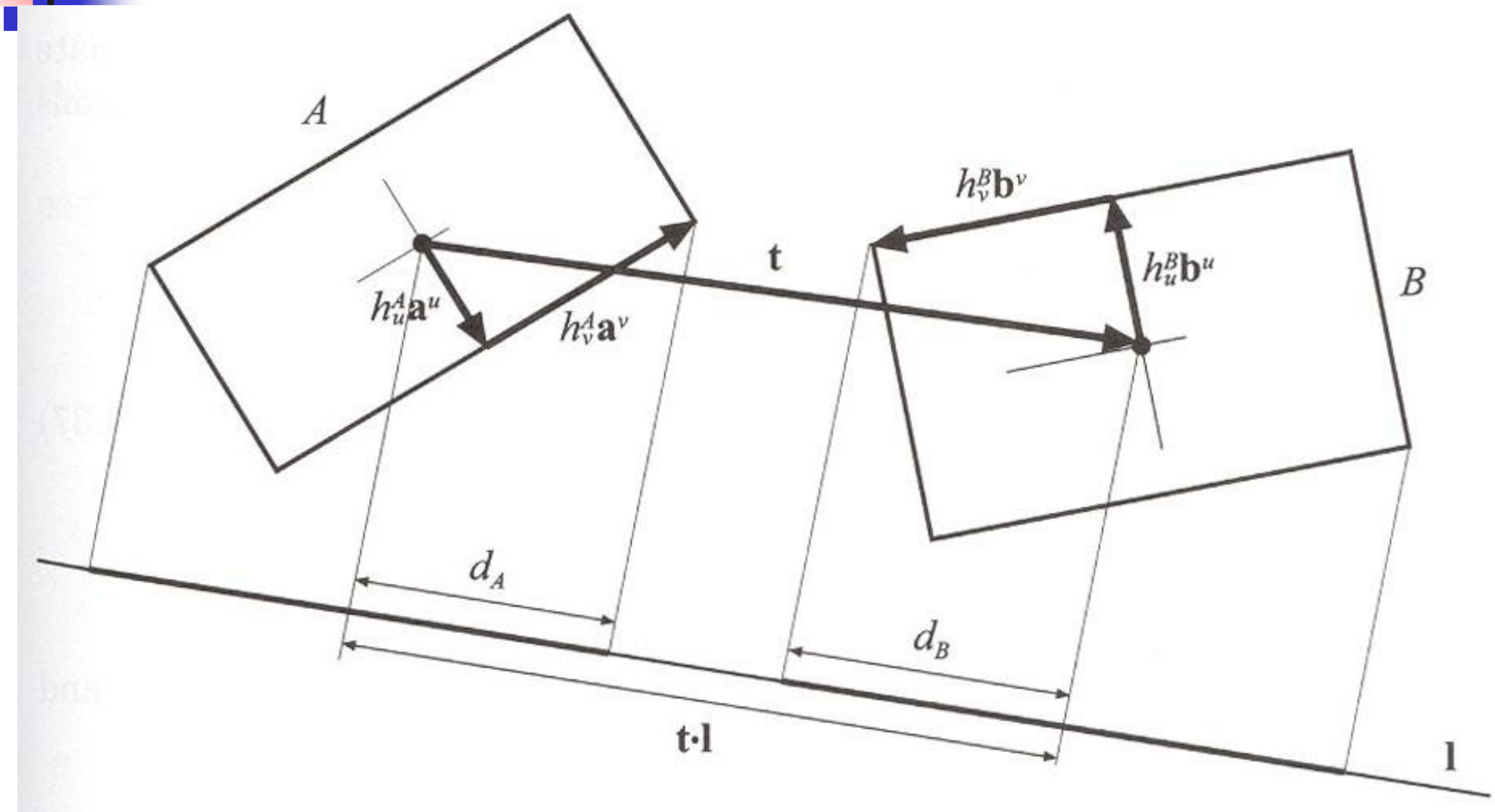


二维形式

为了判断两个OBB是否发生重叠，可使用分离轴定理。因为分离轴应与A和B的表面正交，可选p, q, s, t为分离轴，然后将OBB投影到这些轴上。如果两个OBB在所有这些轴上的投影相互重叠，则两者相交，否则不相交。

因此，为了知道两个OBB是否重叠，只需要找到一条可以使A和B的投影不重叠的轴线即可。在本例中，q轴是唯一能将这两个投影分开的轴。

“OBB/OBB”相交测试



分离轴定理示意。由于A和B在轴线l上的投影的“半长”没有重叠，因此两个OBB不相交。

"OBB/OBB"相交测试

- 对于两个OBB(A和B), 根据分离轴原理，需要对15条轴线进行测试：3条取自A的表面，3条取自B的表面，另外9条取自A和B边的叉乘组合。

- 定理：当且仅当l是一条分离轴时，在这条轴线上的投影才彼此分离，不重叠：

$$|\mathbf{t.l}| > d_A + d_B$$

- 如果在这15个测试中有一个成立，则这两个OBB不相交。

- 最好在开始真正的OBB/OBB测试之前，用包围球进行快速排除。

“线/线”相交测试

- 两条二维直线的相交测试
  - 假设两条直线分别为

$$\mathbf{r}_1(s)=\mathbf{o}_1+s\mathbf{d}_1, \text{和} \mathbf{r}_2(s)=\mathbf{o}_2+s\mathbf{d}_2$$

  - 则交点为：

$$s=\frac{(\mathbf{O}_2-\mathbf{O}_1)\bullet\mathbf{d}_2^{\perp}}{\mathbf{d}_1\bullet\mathbf{d}_2^{\perp}},$$

$$t=\frac{(\mathbf{O}_1-\mathbf{O}_2)\bullet\mathbf{d}_1^{\perp}}{\mathbf{d}_2\bullet\mathbf{d}_1^{\perp}}$$

两条二维线段的相交计算

- 假设两条线段分别为

$$\mathbf{r}_1(s)=\mathbf{p}_1+s(\mathbf{p}_2-\mathbf{p}_1),\text{和}\mathbf{r}_2(s)=\mathbf{q}_1+s(\mathbf{q}_2-\mathbf{q}_1)$$

- 则交点为：

$$s=\frac{-\mathbf{c}\bullet\mathbf{a}^{\perp}}{\mathbf{b}\bullet\mathbf{a}^{\perp}}=\frac{\mathbf{c}\bullet\mathbf{a}^{\perp}}{\mathbf{a}\bullet\mathbf{b}^{\perp}}=\frac{d}{f},$$

$$t=\frac{\mathbf{c}\bullet\mathbf{b}^{\perp}}{\mathbf{a}\bullet\mathbf{b}^{\perp}}=\frac{e}{f},$$

$$\text{其中}\mathbf{a}=\mathbf{q}_2-\mathbf{q}_1,\mathbf{b}=\mathbf{p}_2-\mathbf{p}_1,\mathbf{c}=\mathbf{p}_1-\mathbf{q}_1$$

伪代码

```
if (f > 0)
    if( d < 0 or d > f ) return NO_INTERSECATION
else
    if( d > 0 or d < f ) return NO_INTERSECATION
```

三个平面之间的相交测试

- 给定3个平面，法向量为$\mathbf{n}_i$, 平面上各有一个任意点$\mathbf{p}_i$，则其交点为：

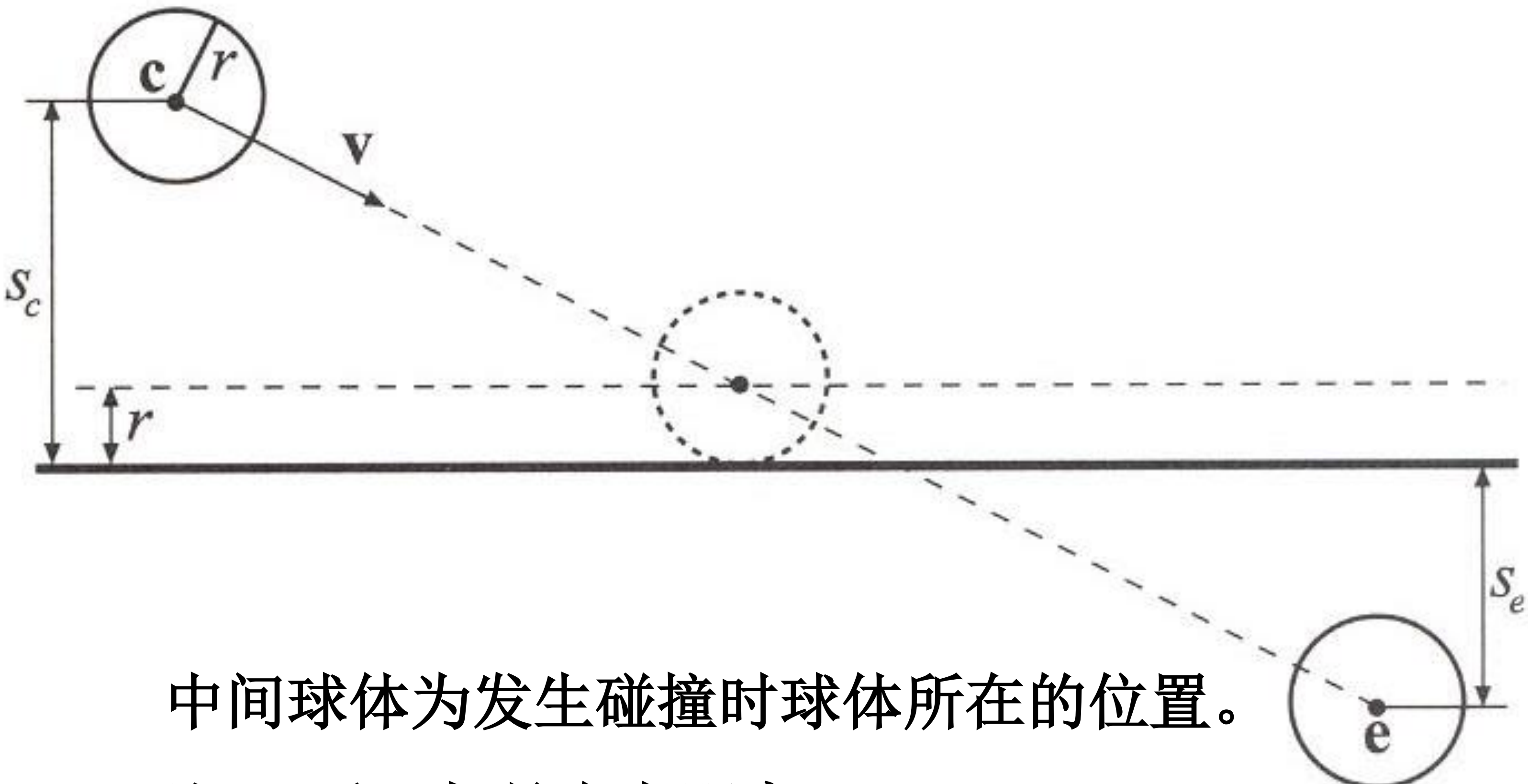
$$\mathbf{p}=\frac{(\mathbf{p}_1\bullet\mathbf{n}_1)(\mathbf{n}_2\times\mathbf{n}_3)+(\mathbf{p}_2\bullet\mathbf{n}_2)(\mathbf{n}_3\times\mathbf{n}_1)+(\mathbf{p}_3\bullet\mathbf{n}_3)(\mathbf{n}_{\text{、}}\times\mathbf{n}_2)}{\begin{vmatrix}\mathbf{n}_1 & \mathbf{n}_2 & \mathbf{n}_3\end{vmatrix}}$$

- 当三个法向行列式为0时，有平面平行，无交。

动态相交测试

- 动机
  - 在动态场景中，物体在时间轴上是连续运动，用离散的方式(t，t+△t, …)，很可能漏掉相交情形。虽然可以通过加密时间间隔进行相交测试，但仍会错过部分相交测试，而且还会增加计算负荷。因此，需要设计动态相交测试来处理这种情况。
- 动态相交测试的运动相对原则
  - 运动是相对的。
  - 假设A和B均在运动，则可看成A是运动的（当然速度需改变），B是静止的。

“球体/平面”动态相交测试



中间球体为发生碰撞时球体所在的位置。

注：s_c和s_e都是有向距离

"球体/平面"动态相交测试

- 假设球体的中心为**c**，半径为r，在△t内物体匀速直线运动，则球心在一条直线上。

- 计算球心和平面之间的有向距离s_c和s_e

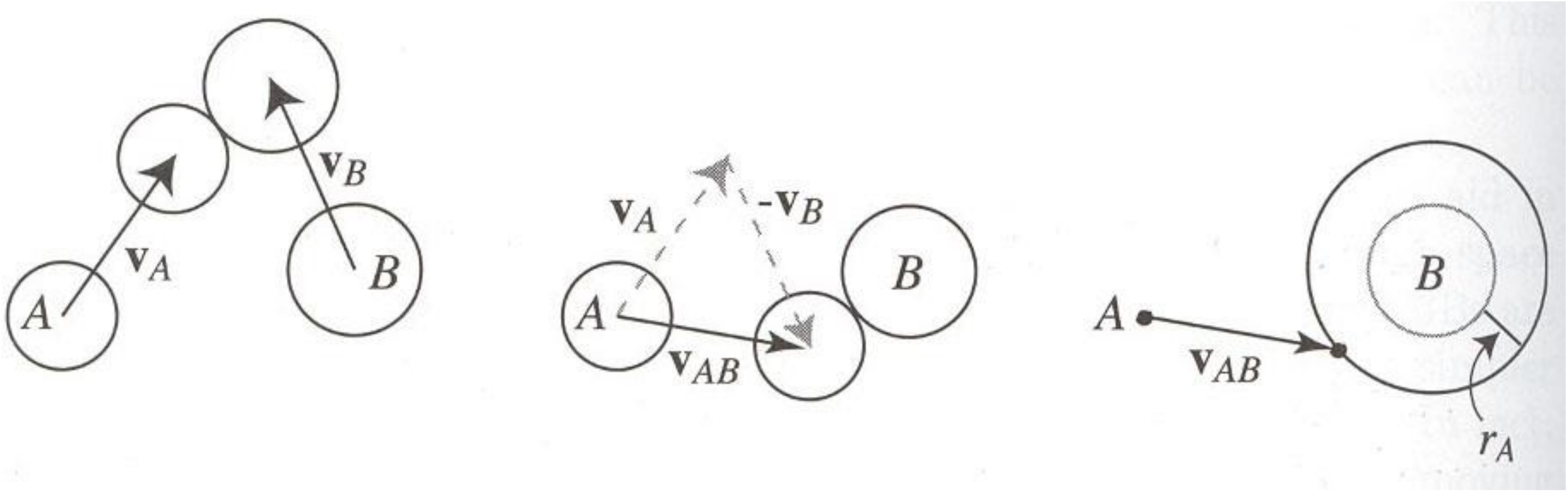
- 如果球心位置在平面的同一侧(即$s_c s_e > 0$)，而且$|s_c| > r$和$|s_e| > r$, 则球体不会与平面相交。
- 否则需要计算相交的准确时刻和球体位置。

$$t = \frac{s_c - r}{s_c - s_e}$$

“球体/球体”相交测试

- 两个球体间的相交测试等同于一条射线与静止球体之间的相交测试。（见下页图）
- 首先，使用相对运动原则，把球体B看成是静止的。
- 这里，可以使一个球在另外一个球的表面移动，从而得到一个新的球体，半径为两个球体的半径之和。
- 假设球体B是新的球体且处于静止状态，半径是两个球体之和，则球体A就是沿着直线运动的一个点。

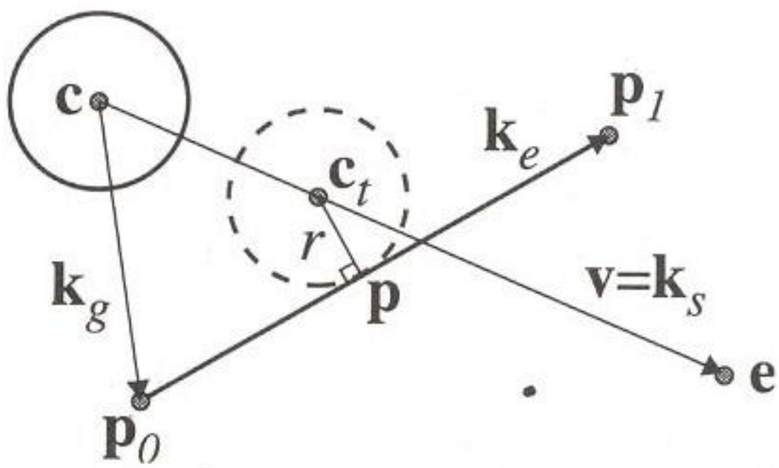
“球体/球体”相交测试



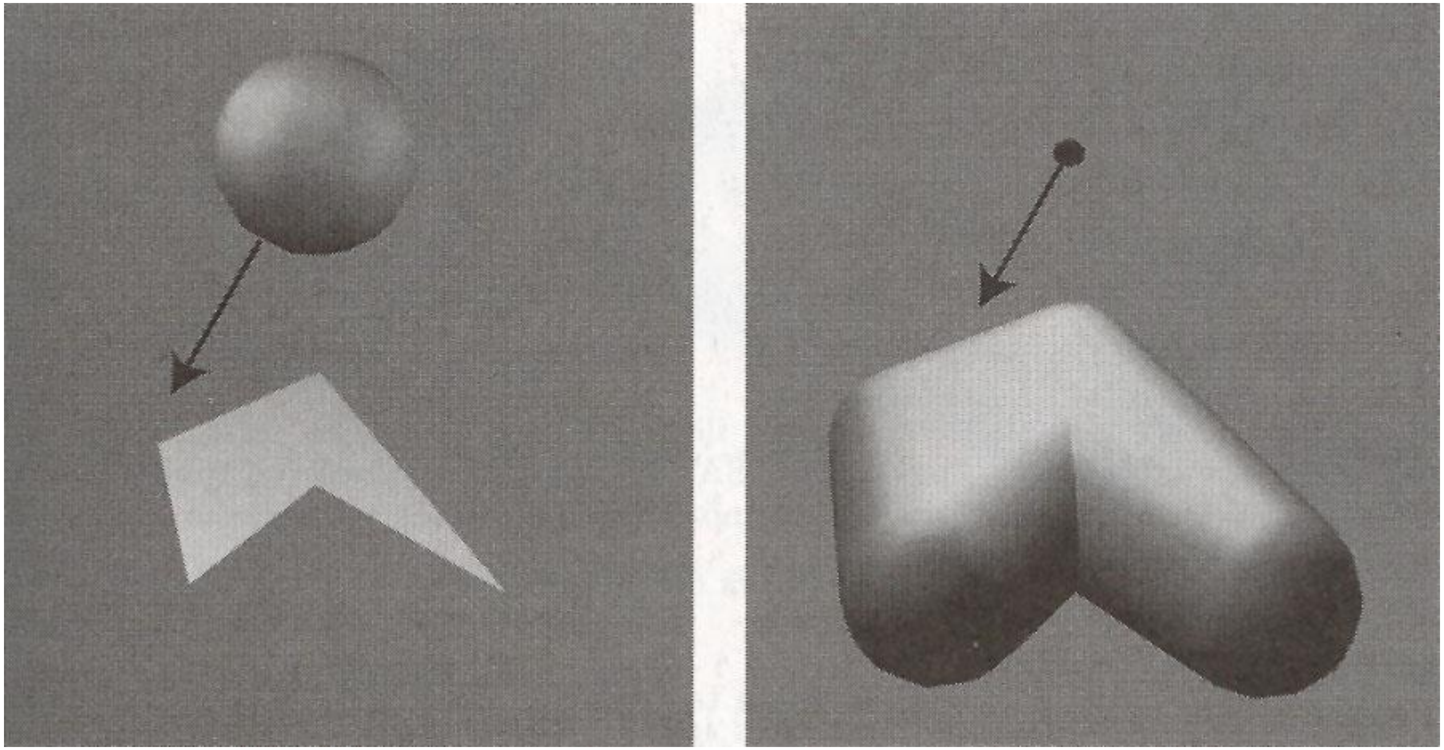
A
$\mathbf{v}_A$
$\mathbf{v}_B$
B
$\mathbf{v}_A$
$-\mathbf{v}_B$
A
B
$\mathbf{v}_{AB}$
A
$\mathbf{v}_{AB}$
B
r_A

“球体/多边形”动态相交测试

- 利用前面的平面与球体的相交方法，测试球体是否和多边形所在的平面相交，如果不相交，则结束。
- 否则，计算球体首次与平面相交的时刻，然后检查交点是否在多边形内部，如果是，则相交；否则需要进一步的检查球体是否与多边形的边或点相交。



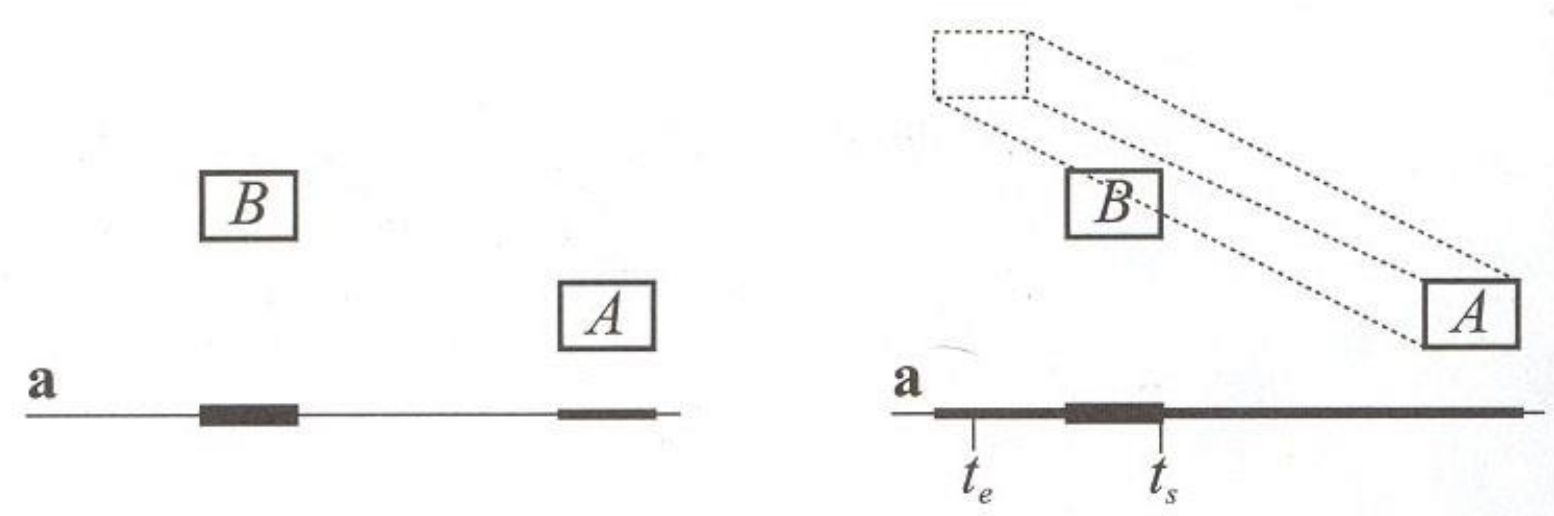
等价处理法



左图：一个球体正向一个多边形移动

右图：一条射线指向一个膨胀的多边形，这两种测试是等价的。

动态分离轴测试方法



左图：对于轴a的分离轴定理(SAT)，其中A和B在该轴上没有重叠。

右图：对于轴a的SAT，其中A是移动的，在运动过程中对A和它在轴a上的区间投影进行跟踪，可以看出两个物体在轴a上是重叠的。

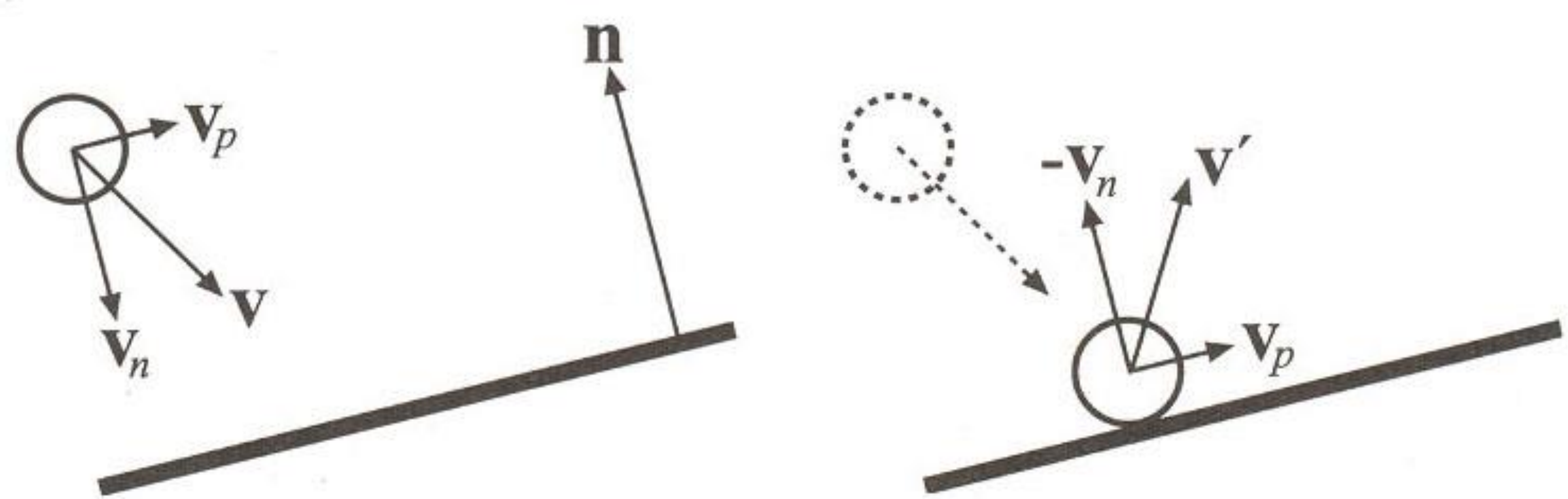
碰撞响应

- 碰撞响应作为一种动作，可以避免物体之间的相互贯穿。

- 球体和平面之间相互碰撞后的响应。
- 假设一个球体向一个平面运动，速度矢量为**v**，平面为∏: **n.x**+d=0，其中**n**为单位化向量。
- 为了计算碰撞响应，可将速度向量表示为：

  $\mathbf{v}=\mathbf{v}_n+\mathbf{v}_p$，其中$\mathbf{v}_n=(\mathbf{v}.\mathbf{n})\mathbf{n}$, $\mathbf{v}_p=\mathbf{v}-\mathbf{v}_n$
- 得到碰撞之后的速度向量**v'**为

$$\mathbf{v}'=\mathbf{v}_p-\mathbf{v}_n$$



完全弹性碰撞。对于非完全弹性碰撞，可以将$-\mathbf{v}_n$长度减小。

「参考资料」

- Tomas Akenine-Möller, Eric Haines, and Naty Hoffman, **Real Time Rendering**, 1045 pages, A.K. Peters Ltd., 3rd edition, ISBN 978-1-56881-424-7, 2008

- http://www.realtimerendering.com/

- **Ming C. Lin的主页** **http://www.cs.unc.edu/~lin/**
  - ✓ **I-COLLIDE**
  - ✓ **RAPID**
  - ✓ **V-COLLIDE**
  - ✓ **PQP**
  - ✓ **H-COLLIDE**
  - ✓ **SWIFT**
  - ✓ **PIVOT**
  - ✓ **SWIFT++**
  - ✓ **DEEP**