计算图形学常用几何工具及数学原理

何小伟
中国科学院软件研究所
2023.4.9

大纲

- 矢量代数（Vector Algebra）
- 张量分析（Tensor Analysis）
- 三维几何图元（Geometric Primitives in 3D）
- 距离计算（Distance in 3D）
- 相交测试（Intersection in 3D）
- 测试场景

矢量代数（Vector Algebra）

- 矢量代数运算

运算类型	表达式
矢量和	$\mathbf{a}+\mathbf{b}$
矢量差	$\mathbf{a}-\mathbf{b}$
矢量点积	$\mathbf{a}\cdot\mathbf{b}$
矢量叉积	$\mathbf{a}\times\mathbf{b}$
三重积	$\mathbf{a}\cdot(\mathbf{b}\times\mathbf{c})$

张量分析（Tensor Analysis）

[张量分析，黄克智]

- 张量定义

	零阶张量（标量）	一阶张量（矢量）	二阶张量（矩阵）
常规标记法	a	$\mathbf{v}=\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$	$\mathbf{A}=\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{11} & a_{22} \end{bmatrix}$
爱因斯坦标记法	a	a_i	a_{ij}

张量分析（Tensor Analysis）

- 张量代数运算

运算类型	表达式
张量和/差	$(\mathbf{A} \pm \mathbf{B})_{ij} = a_{ij} \pm b_{ij}$
标量与张量乘法	$(k\mathbf{A})_{ij} = ka_{ij}$
张量点积	$(\mathbf{A} \cdot \mathbf{B})_{ij} = a_{ik} b_{kj}, k = 0,1,2 \ldots$
张量转置	$\left(\mathbf{A}^{T}\right)_{ij} = a_{ji}$
张量缩并	$\mathbf{A} : \mathbf{B} = a_{ij} b_{ij}$
张量并矢	$(\mathbf{a} \otimes \mathbf{b})_{ij} = a_i b_j$

$\otimes$

三维几何图元

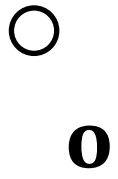
- 点
- 直线、射线、线段
- 平面、三角形
- 球、胶囊
- 四面体
- AABB、OBB

src/Core/Primitive/Primitive3D.h

三维几何图元

- 点

```
template<typename Real>
class TPoint3D
{
    Coord3D origin;
};
```

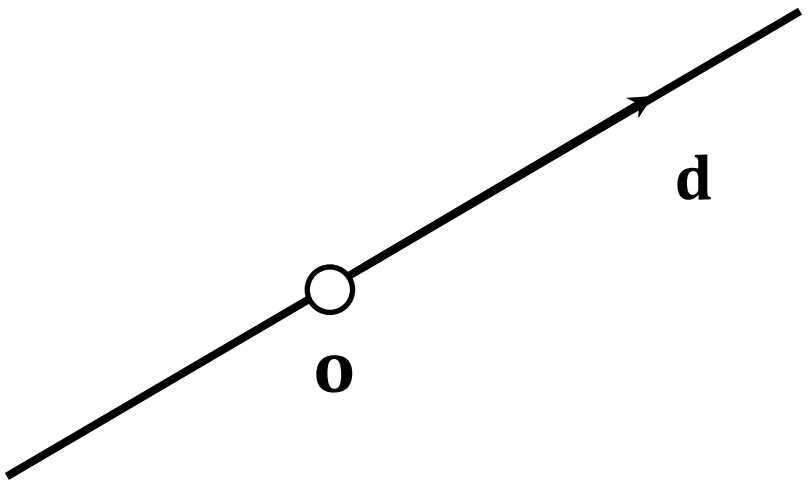


Parametric form: $\mathbf{o} \equiv (x, y, z)$

三维几何图元

- 直线

```
template<typename Real>
class TLine3D
{
    Coord3D origin;
    Coord3D direction;
};
```

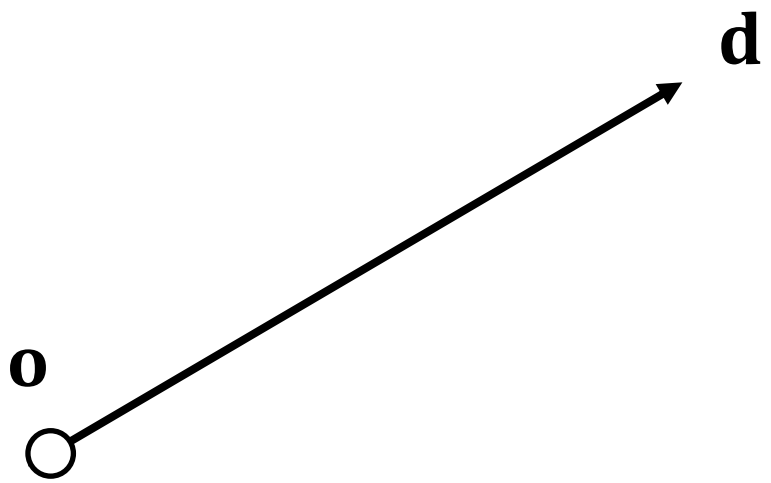


Parametric form: $\mathbf{x}(t) = \mathbf{o} + t\mathbf{d}, t \in (-\infty, \infty)$

三维几何图元

- 射线

```
template<typename Real>
class TRay3D
{
    Coord3D origin;
    Coord3D direction;
};
```

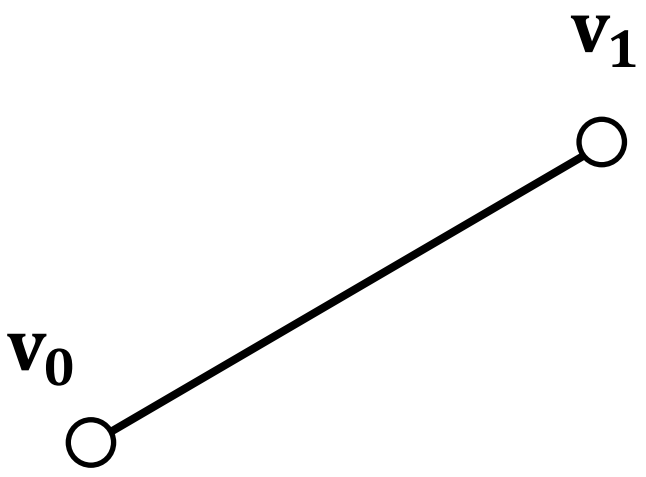


Parametric form: $\mathbf{x}(t) = \mathbf{o} + t\mathbf{d}, t \in [0, \infty)$

三维几何图元

- 线段

```
template<typename Real>
class TSegment3D
{
   Coord3D v0;
   Coord3D v1;
};
```

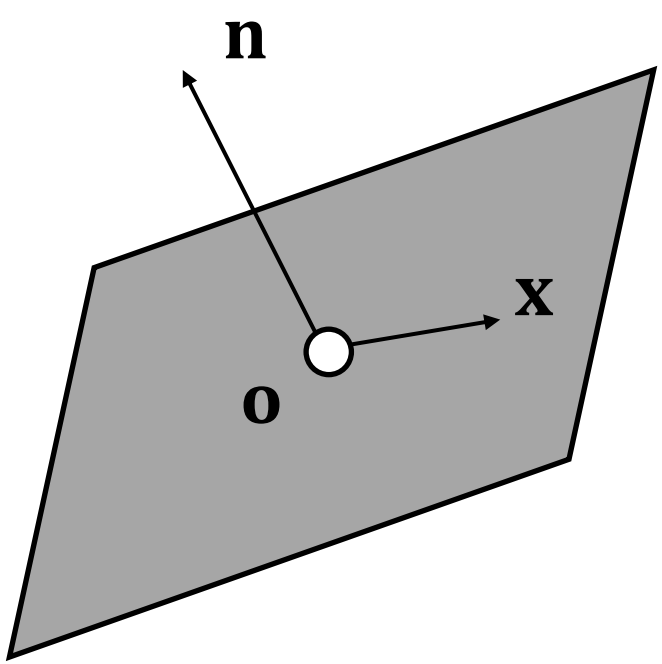


Parametric form: $\mathbf{x}(t) = (1-t)\mathbf{v_0} + t\mathbf{v_1}, t \in [0,1]$

三维几何图元

- 平面

```
template<typename Real>
class TPlane3D
{
    Coord3D origin;
    Coord3D normal;
};
```



$$(\mathbf{x}-\mathbf{o})\cdot\mathbf{n}=\mathbf{0}$$

Parametric form:

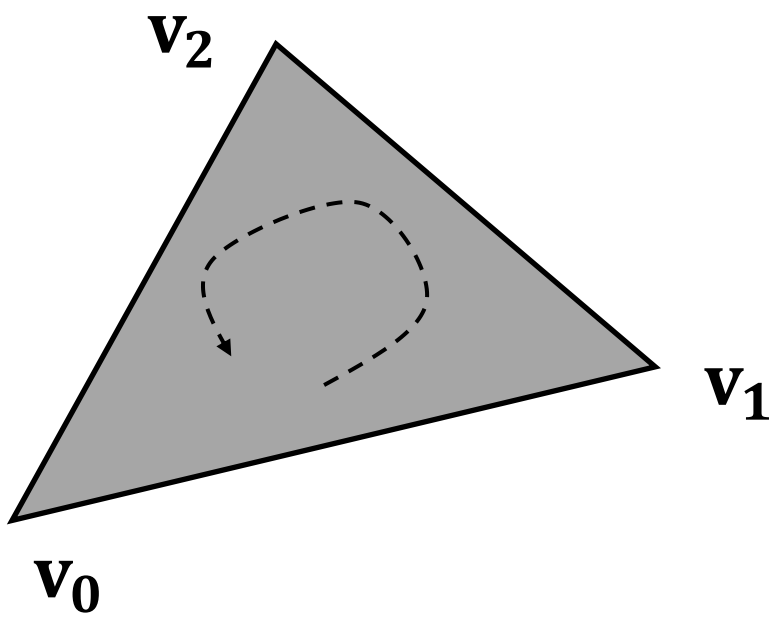
$$ax+by+cz+d=0$$

$$\mathbf{n}=(a,b,c)$$

三维几何图元

- 三角形

```
template<typename Real>
class TTriangle3D
{
   Coord3D v[3];
};
```



Parametric form:

$$\mathbf{x}(s,t) = \mathbf{v_0} + s\mathbf{e_0} + t\mathbf{e_1}$$

$$\mathbf{e_0} = \mathbf{v_1} - \mathbf{v_0}$$

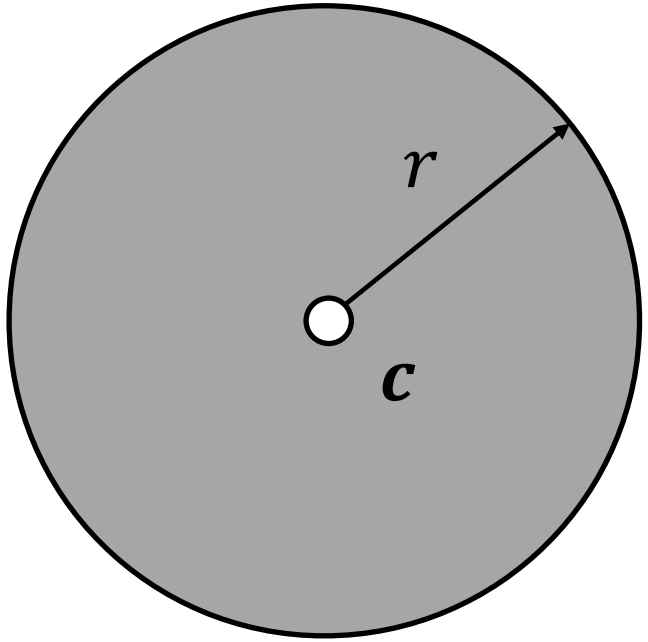
$$\mathbf{e_1} = \mathbf{v_2} - \mathbf{v_0}$$

$$s + t, s, t \in [0, 1]$$

三维几何图元

- 球

```
template<typename Real>
class TSphere3D
{
    Coord3D center;
    Real radius;
};
```

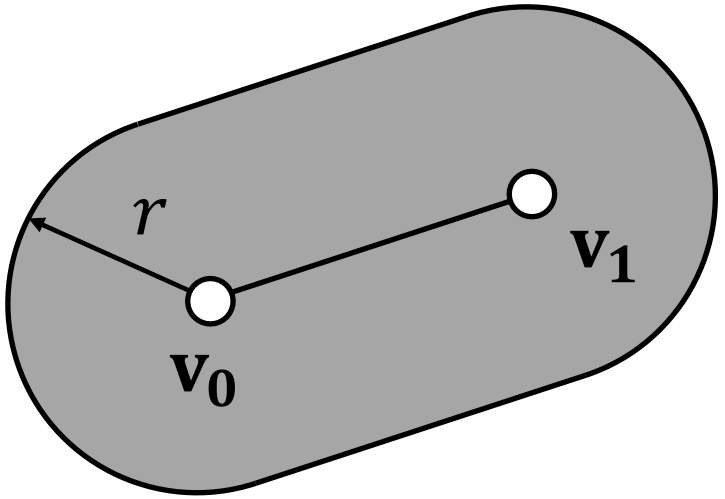


Parametric form: $\|\mathbf{x}-\mathbf{c}\|^2 \le r^2$

三维几何图元

- 胶囊

```
template<typename Real>
class TCapsule3D
{
    TSegment<Real> segment;
    Real radius;
};
```

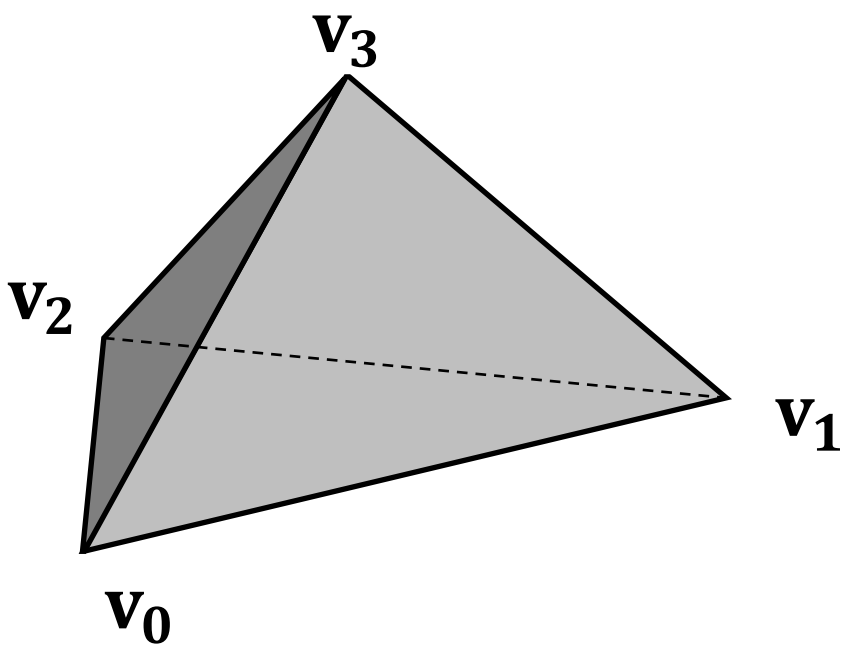


Parametric form: $\|\mathbf{x} - \overline{\mathbf{v_0}\mathbf{v_1}}\|^2 \leq r^2$

三维几何图元

- 四面体

```
template<typename Real>
class TTet3D
{
   Coord3D v[4];
};
```



Parametric form:

$$\mathbf{x}(u,v,w) = \mathbf{v}_0 + u\mathbf{e}_0 + v\mathbf{e}_1 + w\mathbf{e}_2$$

$$\mathbf{e}_0 = \mathbf{v}_1 - \mathbf{v}_0$$

$$\mathbf{e}_1 = \mathbf{v}_2 - \mathbf{v}_0$$

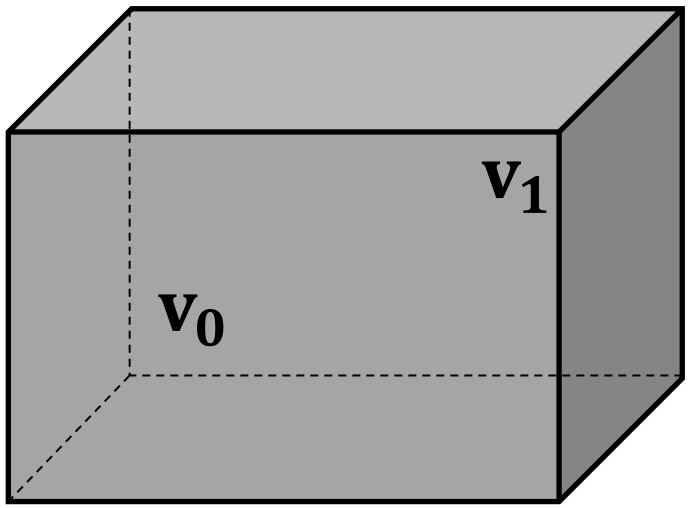
$$\mathbf{e}_2 = \mathbf{v}_3 - \mathbf{v}_0$$

$$u + v + w, u, v, w \in [0,1]$$

三维几何图元

- Axis Aligned Bounding Box(AABB)

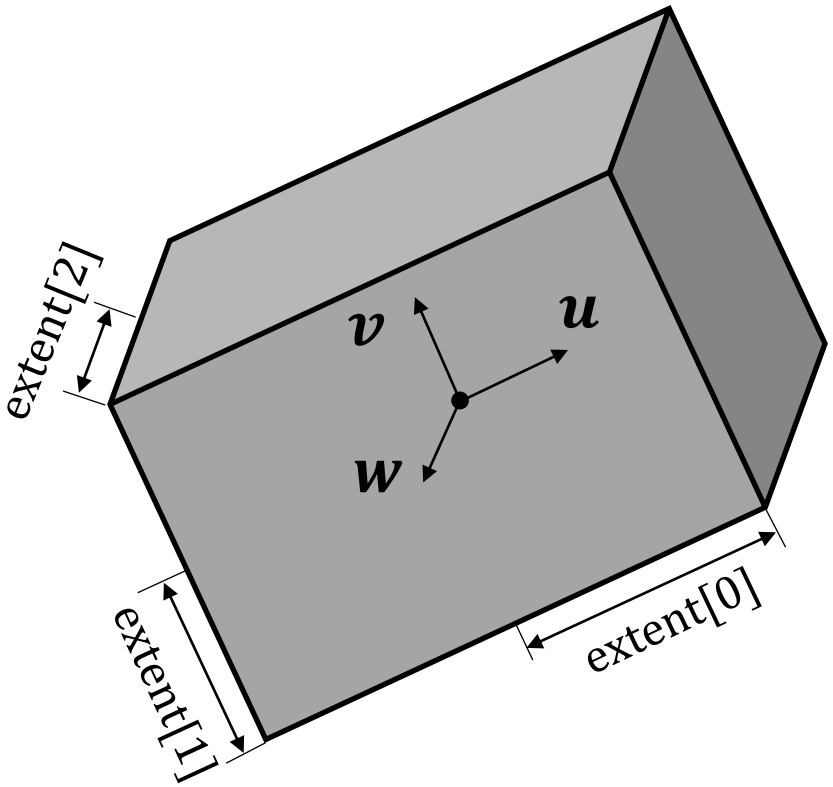
```
template<typename Real>
class TAlignedBox3D
{
    Coord3D v0;
    Coord3D v1;
};
```



三维几何图元

- Oriented Bounding Box(OBB)

```
template<typename Real>
class TOrientedBox3D
{
    Coord3D center;
    Coord3D u, v, w;
    Coord3D extent;
};
```



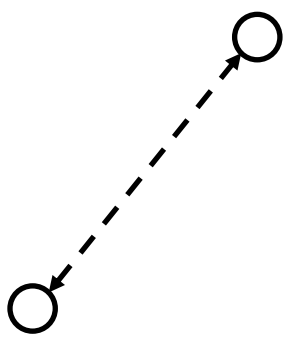
距离计算

- 点到点距离
- 点到直线/射线/线段的距离
- 点到平面/三角形的距离
- 点到球/胶囊的距离
- 点到四面体的距离
- 点到AABB/OBB的距离

距离计算

- 点到点距离

```
template<typename Real>
Real TPoint3D<Real>::distance(const
        TPoint3D<Real>& pt) const
{
        return (origin - pt.origin).norm();
}
```



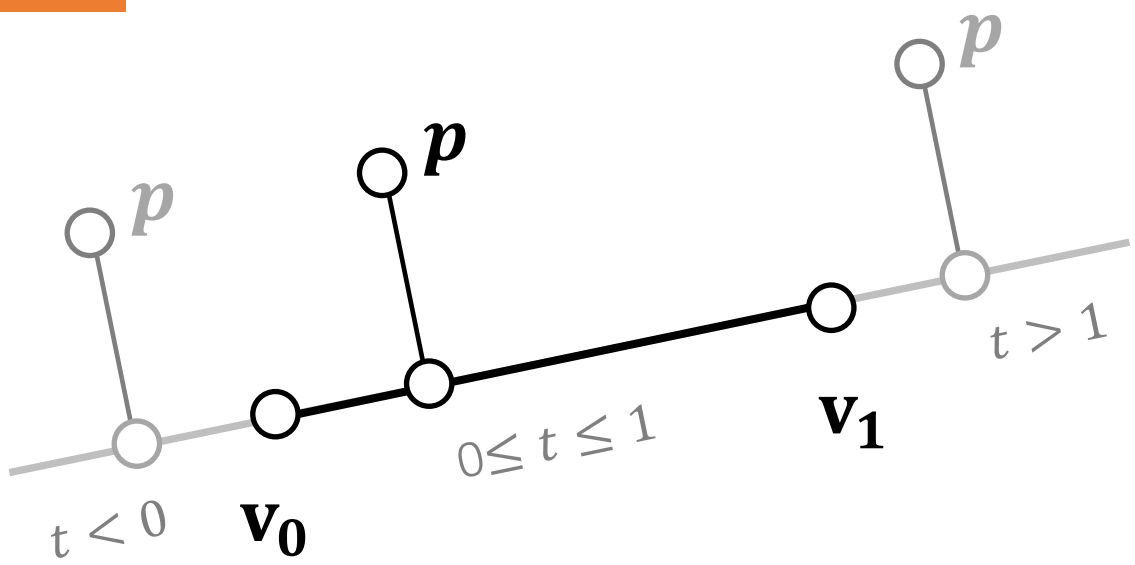
$$\mathbf{o_1} \equiv (x_1, y_1, z_1)$$

$$\mathbf{o_0} \equiv (x_0, y_0, z_0)$$

$$d = \|\mathbf{o_0} - \mathbf{o_1}\|$$

距离计算

- 点到线性结构的距离

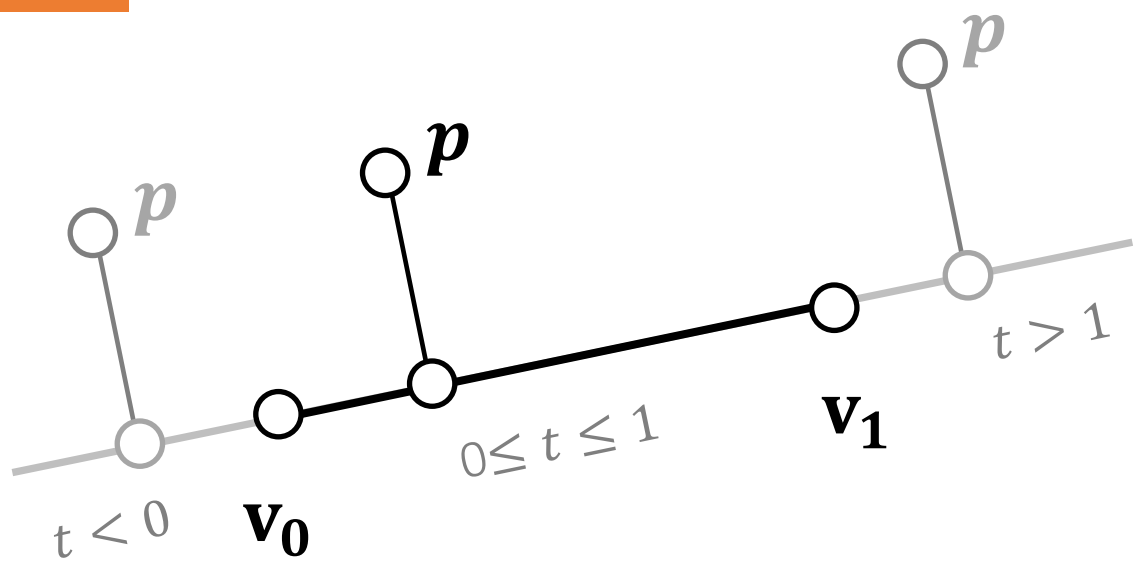


$$t = \frac{(\boldsymbol{p} - \mathbf{v_0}) \cdot (\mathbf{v_1} - \mathbf{v_0})}{\|\mathbf{v_1} - \mathbf{v_0}\|^2}$$

距离计算

- 点到直线的距离

```
template<typename Real>
Real  TPoint3D<Real>::distance(const
        TLine3D<Real>& line) const
{
        Coord3D u = origin - line.origin;
        Real tNum = u.dot(line.direction);
        Real a = line.direction.normSquared();
        Real t = a < REAL_EPSILON_SQUARED ?
                0 : tNum / a;

        Coord3D pj = line.origin +
                t * line.direction);

        return (origin - pj).norm();
}
```



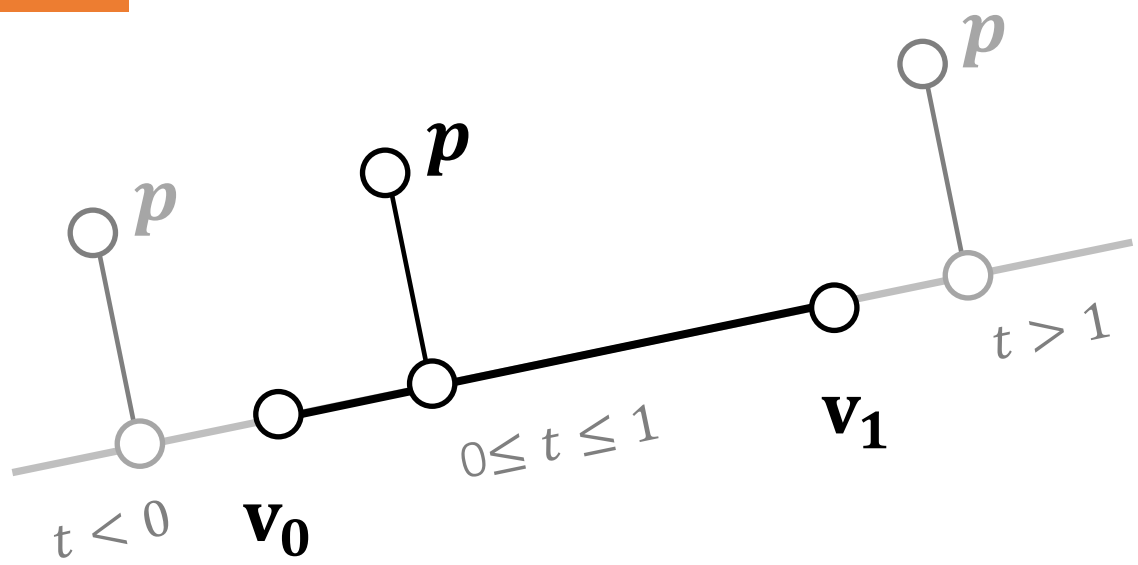
$$\mathbf{x}(t) = \mathbf{v_0} + t(\mathbf{v_1} - \mathbf{v_0})$$

$$\begin{cases} \text{直线：} & -\infty < t < \infty \\ \text{射线：} & 0 \leq t \\ \text{线段：} & 0 \leq t \leq 1 \end{cases}$$

距离计算

- 点到射线的距离

```
template<typename Real>
Real  TPoint3D<Real>::distance(const
        TRay3D<Real>& ray) const
{
        Coord3D u = origin - ray.origin;
        Real tNum = u.dot(ray.direction);
        Real a = ray.direction.normSquared();
        Real t = a < REAL_EPSILON_SQUARED ?
                0 : tNum / a;

        t = t < 0 ? 0 : t;

        Coord3D pj = ray.origin +
                t * ray.direction);

        return (origin - pj).norm();
}
```

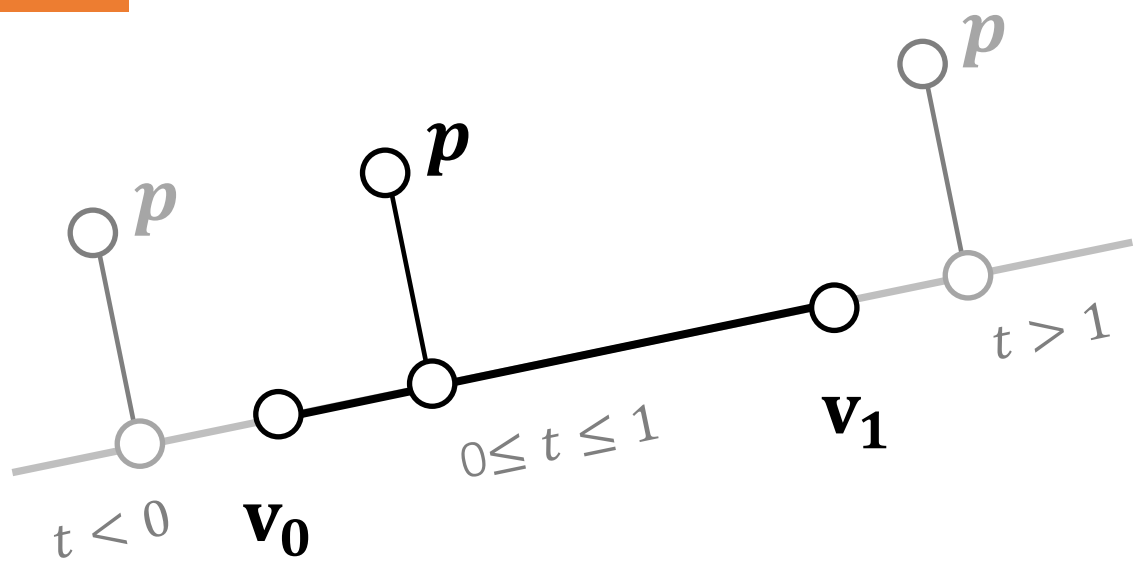


$$t = max\left(\frac{(\boldsymbol{p}-\mathbf{v_0})\cdot(\mathbf{v_1}-\mathbf{v_0})}{\|\mathbf{v_1}-\mathbf{v_0}\|^2}, 0\right)$$

距离计算

- 点到线段的距离

```
template<typename Real>
Real  TPoint3D<Real>::distance(const
        TSegment3D<Real>& line) const
{
        Coord3D l = origin - segment.v0;
        Coord3D dir = segment.v1 - segment.v0;
        if (dir.normSquared() <
                REAL_EPSILON_SQUARED){
                return TPoint3D<Real>(segment.v0);
        }
        Real t = l.dot(dir) / dir.normSquared();

        Coord3D q = segment.v0 + t * dir;
        q = t < 0 ? segment.v0 : q;
        q = t > 1 ? segment.v1 : q;

        return (origin - q).norm();
}
```

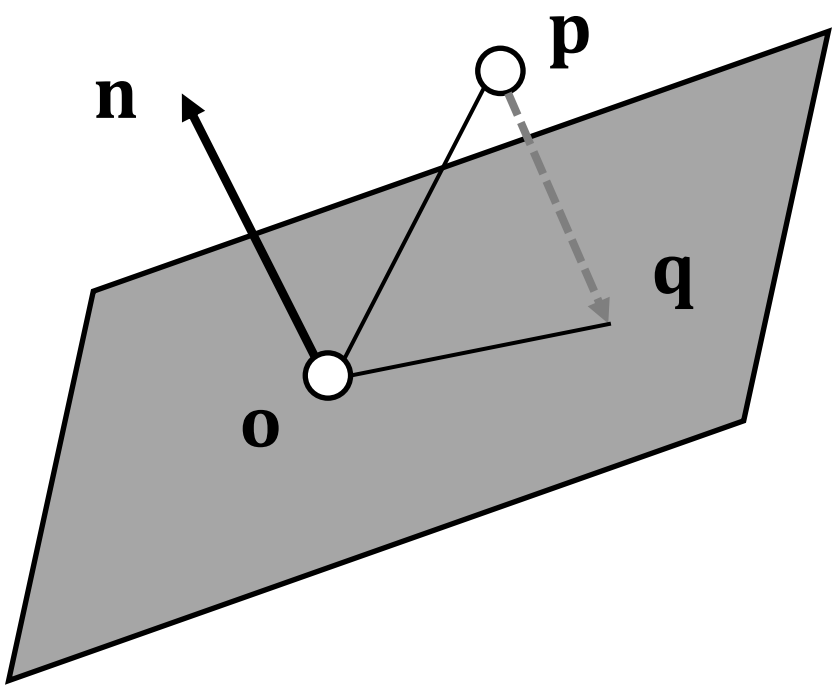


$$t = clamp\left(\frac{(\boldsymbol{p} - \mathbf{v_0}) \cdot (\mathbf{v_1} - \mathbf{v_0})}{\|\mathbf{v_1} - \mathbf{v_0}\|^2}, 0, 1\right)$$

距离计算

- 点到平面的距离

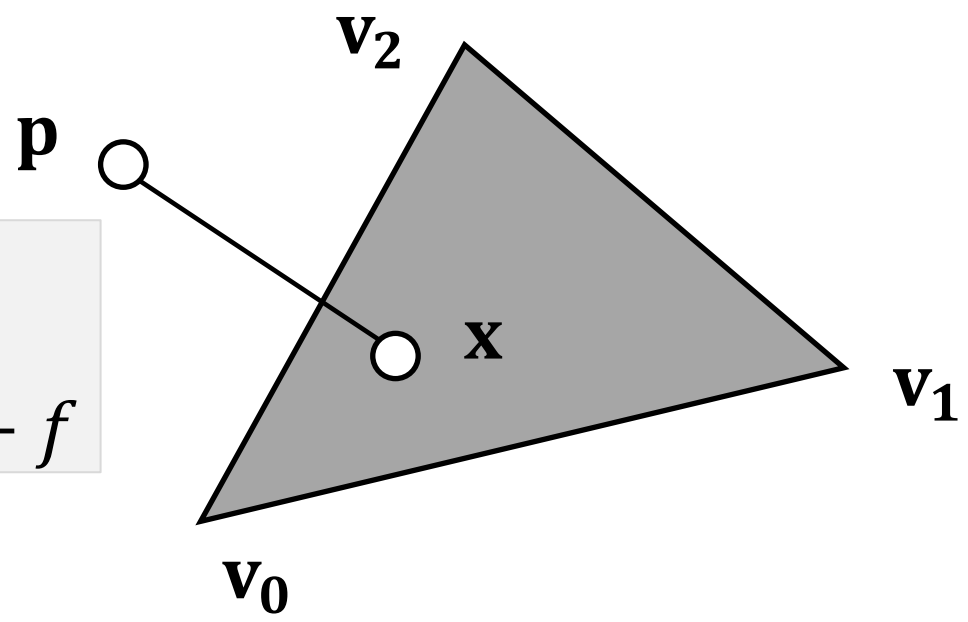
```
template<typename Real>
Real  TPoint3D<Real>::distance(const
        TPlane3D<Real>& plane) const
{
        Real t = (origin -
                plane.origin).dot(plane.normal);

        Real n2 = plane.normal.normSquared();

        Coord3D q = origin - t / n2 *
                plane.normal;

        Real sign = (origin - q).dot(plane.normal)
                < Real(0) ? Real(-1) : Real(1);

        return sign*(origin - q).norm();
}
```



$$\mathbf{q} = \boldsymbol{p} - \frac{(\boldsymbol{o} - \boldsymbol{p}) \cdot \boldsymbol{n}}{\|\boldsymbol{n}\|^2}$$

距离计算

- 点到三角形的距离



$$Q(s,t) = \|\mathbf{x}(s,t) - \mathbf{p}\|^2$$
$$= as^2 + 2bst + ct^2 + 2ds + 2et + f$$

$$a = \mathbf{e_0} \cdot \mathbf{e_0}$$
$$b = \mathbf{e_0} \cdot \mathbf{e_1}$$
$$c = \mathbf{e_1} \cdot \mathbf{e_1}$$
$$d = \mathbf{e_0} \cdot (\mathbf{v_0} - \mathbf{p})$$
$$e = -\mathbf{e_1} \cdot (\mathbf{v_0} - \mathbf{p})$$
$$f = (\mathbf{v_0} - \mathbf{p}) \cdot (\mathbf{v_0} - \mathbf{p})$$

$$\mathbf{x}(s,t) = \mathbf{v_0} + s\mathbf{e_0} + t\mathbf{e_1}$$
$$\mathbf{e_0} = \mathbf{v_1} - \mathbf{v_0}$$
$$\mathbf{e_1} = \mathbf{v_2} - \mathbf{v_0}$$

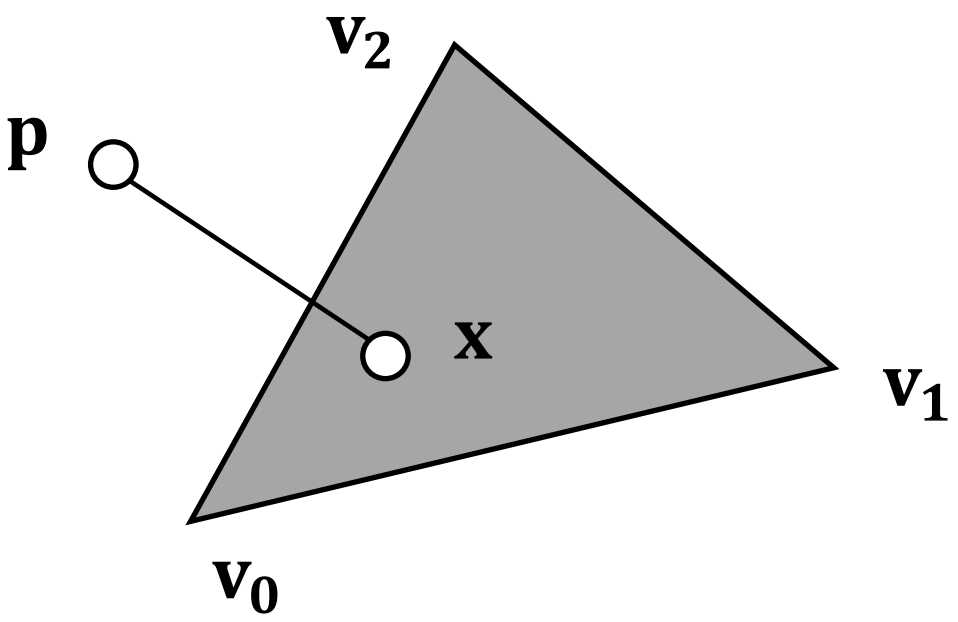
$$s + t = 1, s, t \in [0,1]$$

距离计算

- 点到三角形的距离

$$\nabla Q = \begin{cases} as + bt + d \\ bs + ct + e \end{cases} = 0$$

$$s = \frac{be - cd}{ac - b^2}, t = \frac{bd - ae}{ac - b^2}$$



$$\mathbf{x}(s,t) = \mathbf{v_0} + s\mathbf{e_0} + t\mathbf{e_1}$$

$$\mathbf{e_0} = \mathbf{v_1} - \mathbf{v_0}$$

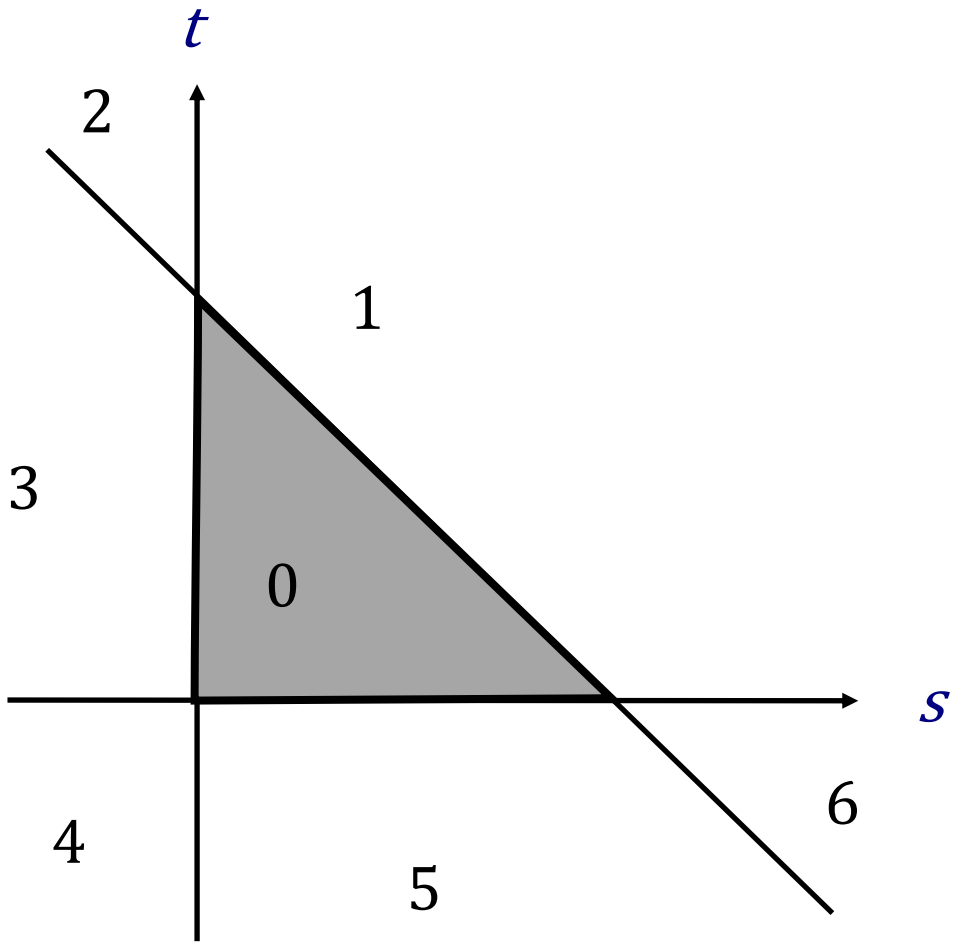
$$\mathbf{e_1} = \mathbf{v_2} - \mathbf{v_0}$$

$$s + t = 1, s, t \in [0, 1]$$

距离计算

- 点到三角形的距离

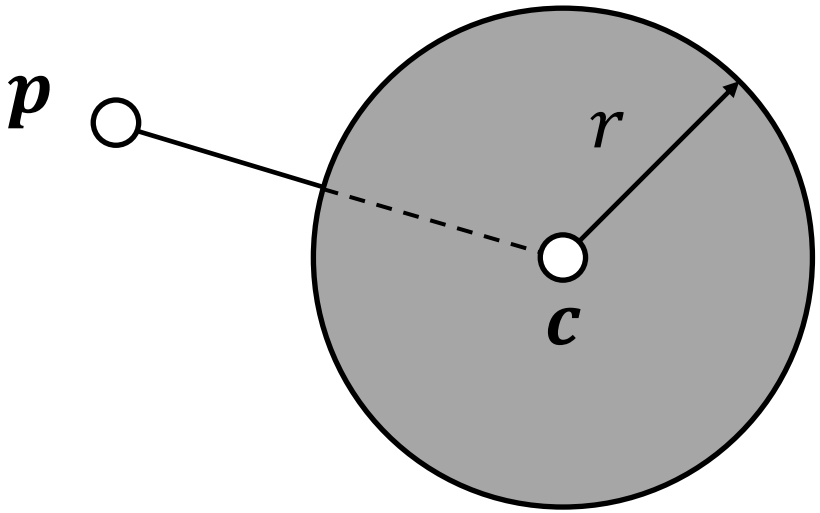
```
if (s + t <=1){
    if (s < 0){
        if (t < 0){//region 4}
        else      {// region 3}
    }
    else{
        if (t < 0){//region 5}
        else      {//region 0}
    }
}
else{
    if (s < 0)      {//region 2}
    else if (t < 0) {//region 6}
    else            {//region 1}
}
```



距离计算

- 点到球的距离

```
template<typename Real>
Real  TPoint3D<Real>::distance(const
        TSphere3D<Real>& sphere) const
{
        return (origin - sphere.center).norm() -
                sphere.radius;
}
```

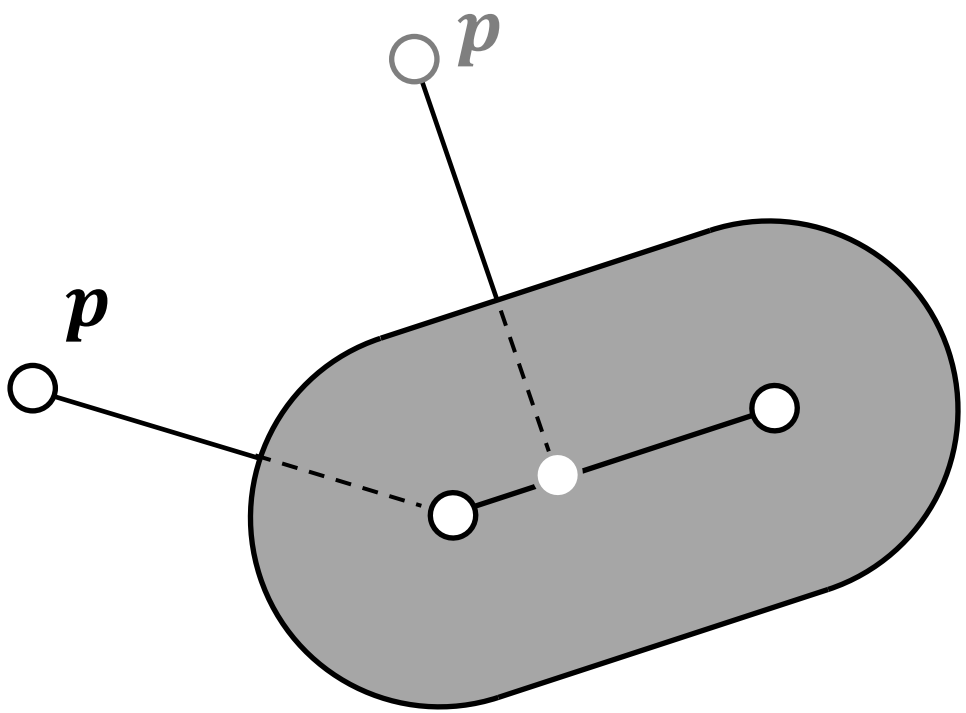


$$d = \|\boldsymbol{p} - \boldsymbol{c}\| - r$$

距离计算

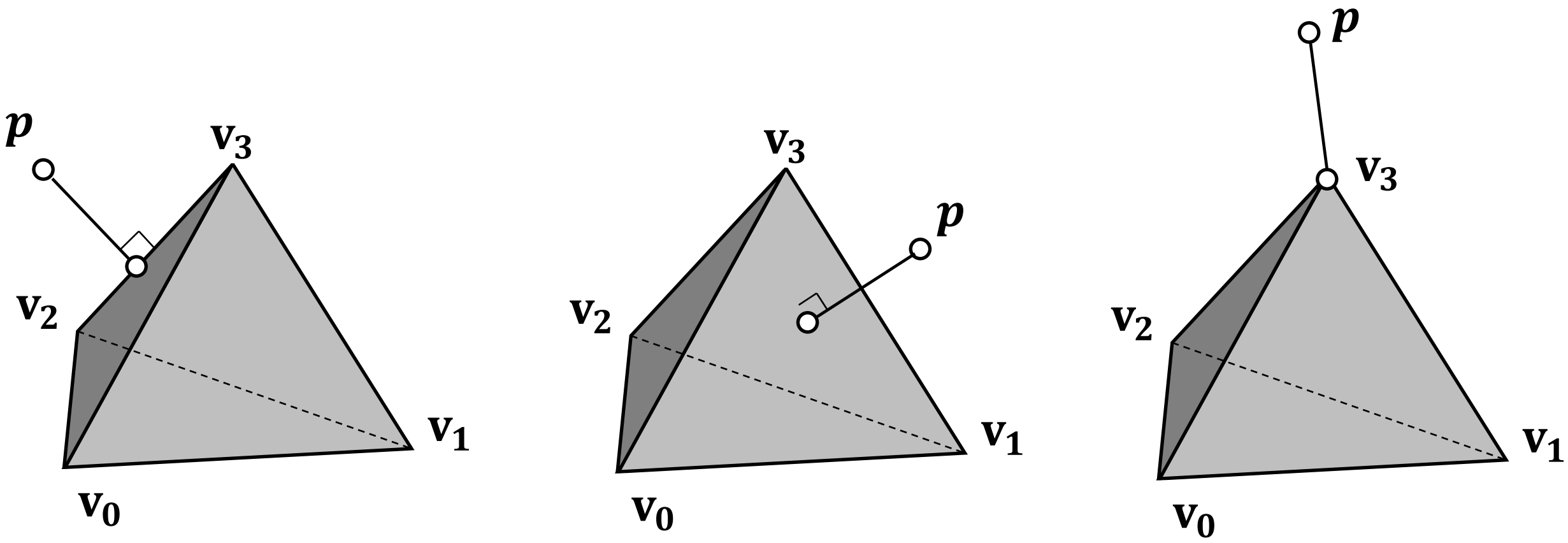
- 点到胶囊体的距离

```
template<typename Real>
Real  TPoint3D<Real>::distance(const
        TCapsule3D<Real>& capsule) const
{
        Bool bInside;
        Real d = (origin - project(capsule,
                bInside).origin).norm();
        return bInside == true ? -d : d;
}
```



距离计算

- 点到四面体的距离



距离计算

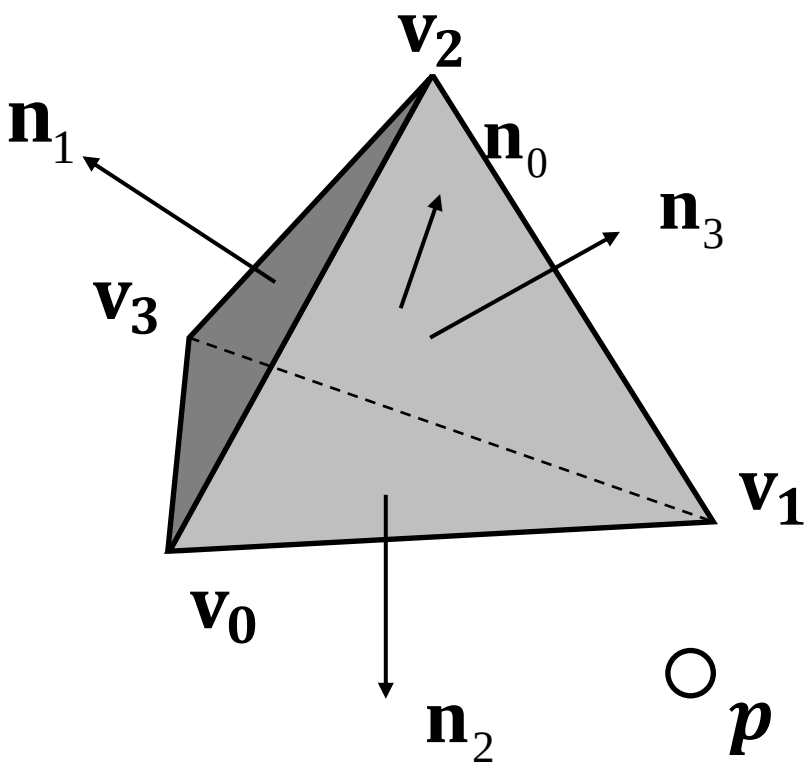
- 点到四面体的距离

$$\mathbf{n}_0 = (\mathbf{v}_2 - \mathbf{v}_3) \times (\mathbf{v}_1 - \mathbf{v}_3)$$

$$\mathbf{n}_1 = (\mathbf{v}_3 - \mathbf{v}_2) \times (\mathbf{v}_0 - \mathbf{v}_2)$$

$$\mathbf{n}_2 = (\mathbf{v}_0 - \mathbf{v}_1) \times (\mathbf{v}_3 - \mathbf{v}_1)$$

$$\mathbf{n}_3 = (\mathbf{v}_1 - \mathbf{v}_0) \times (\mathbf{v}_2 - \mathbf{v}_0)$$

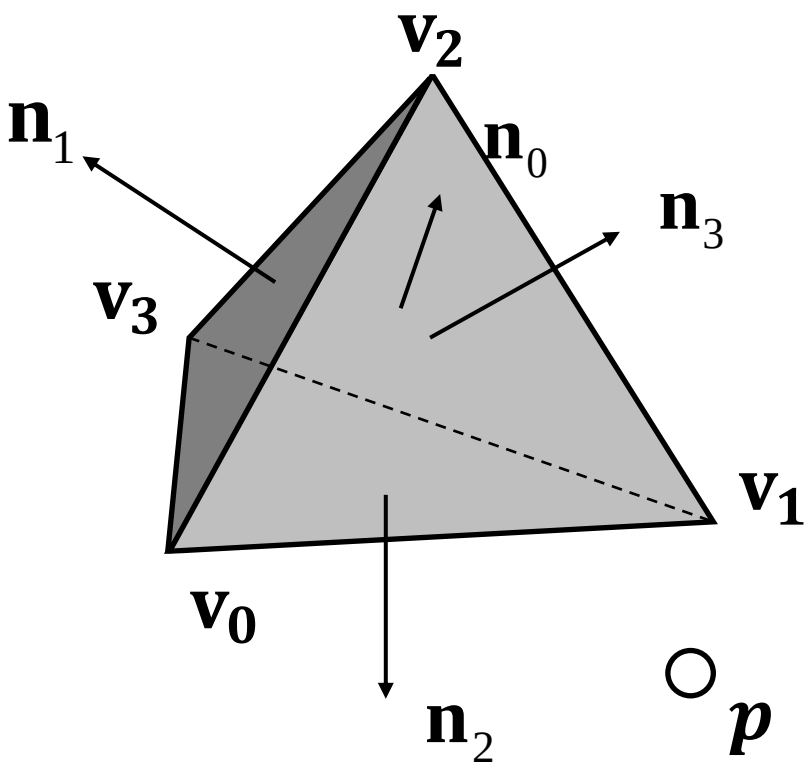


$$b_i = \mathbf{n}_3 \cdot (\mathbf{p} - \mathbf{v}_{3-i}) \le 0$$

距离计算

- 点到四面体的距离

```
template<typename Real>
Real TPoint3D<Real>::distance(const
        TTet3D<Real>& tet) const
{
        Bool bInside = inside(tet);
        TPoint3D<Real> closestPt;
        Real minDist = REAL_MAX;
        for (int i = 0; i < 4; i++)
        {
                TPoint3D<Real> q = project(tet.face(i));
                Real d = (origin - q.origin).normSquared();
                if (d < minDist)
                {
                        minDist = d;
                        closestPt = q;
                }
        }
        Real d = (origin - closestPt.origin).norm();
        return bInside == true ? -d : d;
}
```

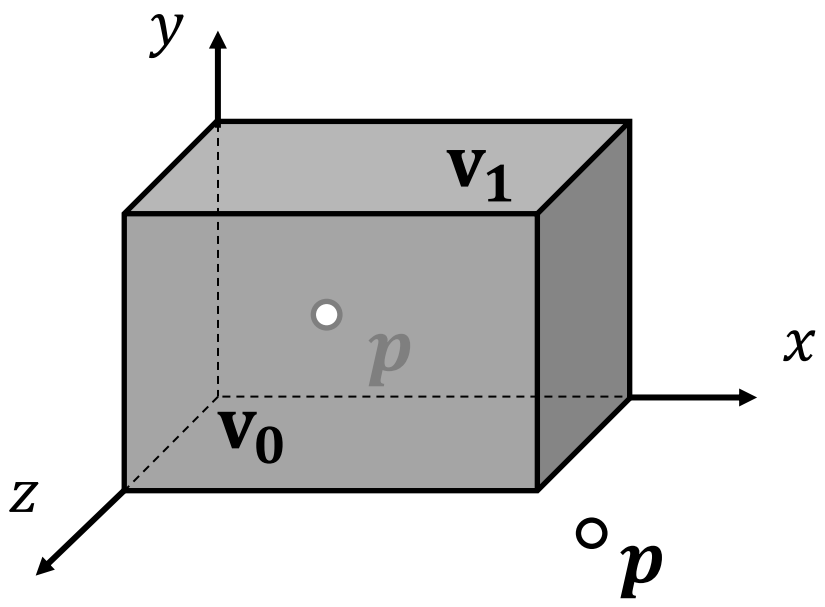


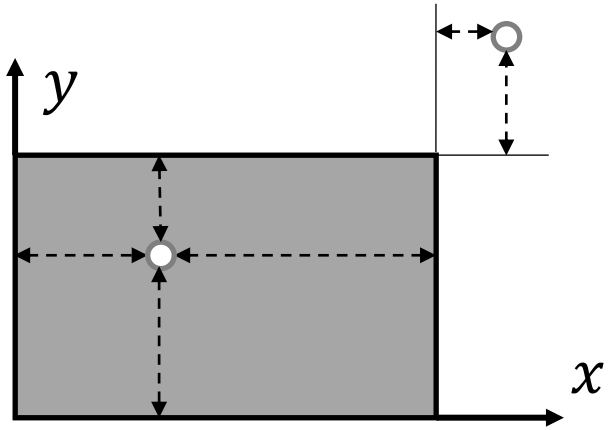
$$b_i = \mathbf{n}_3 \cdot \left(\mathbf{p} - \mathbf{v}_{3-i}\right) \le 0$$

距离计算

- 点到AABB的距离

思路同上

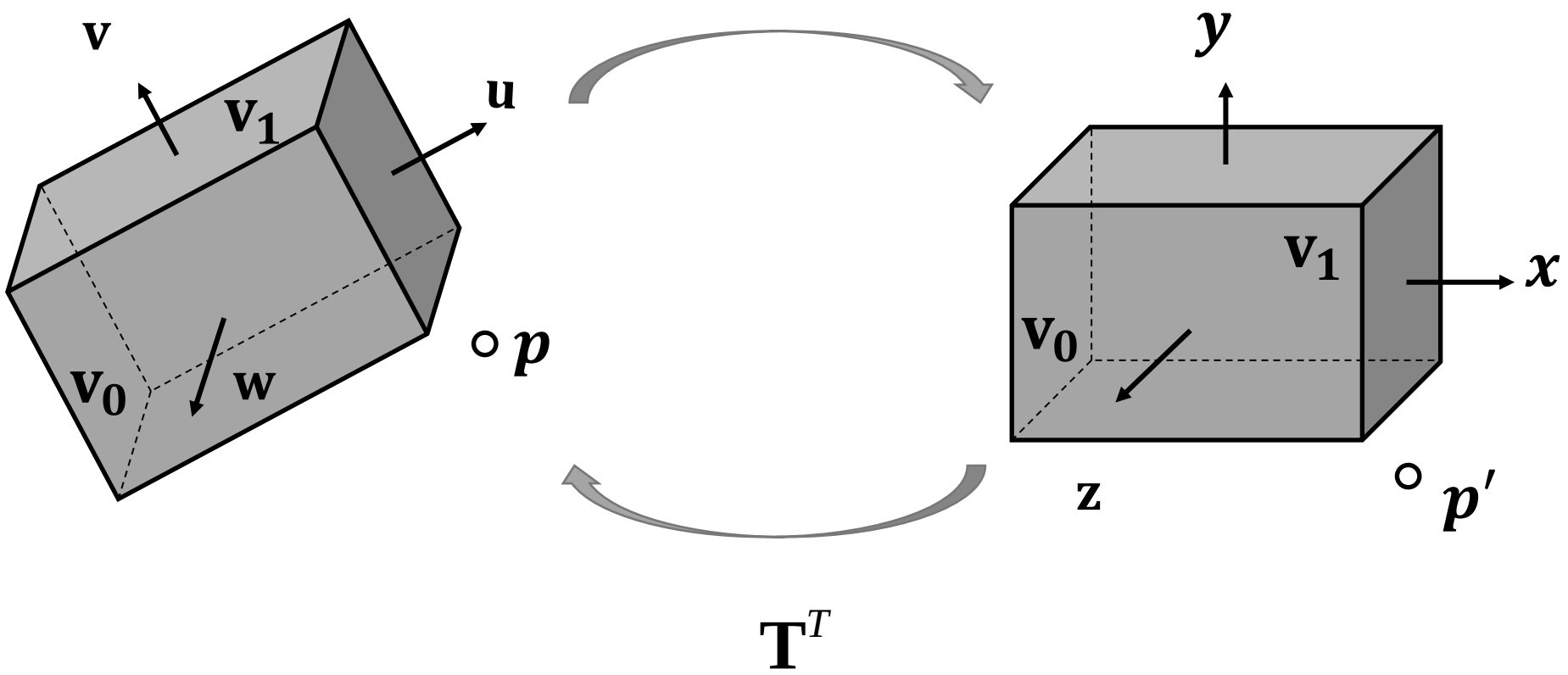




距离计算

- 点到OBB的距离

$$\mathbf{T} = \begin{bmatrix} \mathbf{u}^T & \mathbf{v}^T & \mathbf{w}^T & 0 \\ c_x & c_y & c_y & 1 \end{bmatrix}$$



相交测试

- 线性结构与平面的相交
- 线性结构与球的相交
- 线性结构与三角形的相交
- 线性结构与四面体的相交
- 线性结构与AABB/OBB的相交

相交测试

- 线性结构与平面的相交

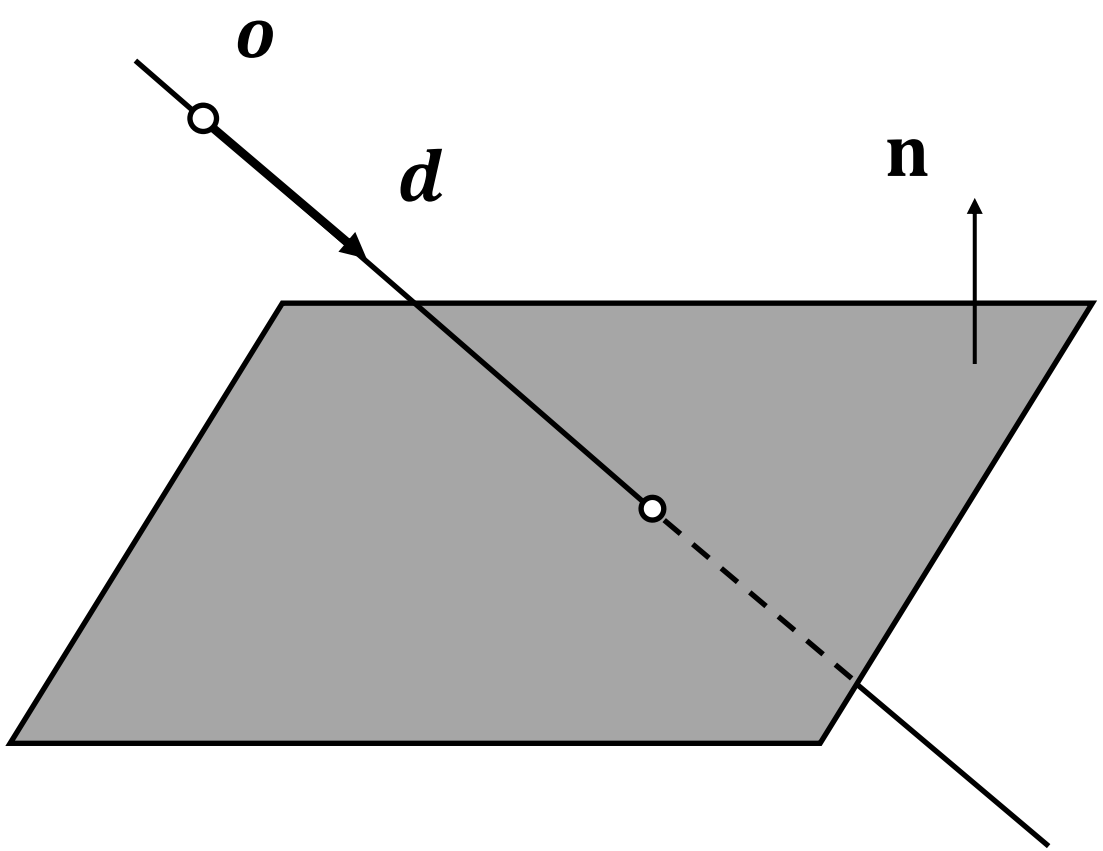
直线：$\mathbf{x}(t) = \mathbf{o} + t\mathbf{d}, t \in (-\infty, \infty)$

平面：$ax + by + cz + d = 0$

$$a(o_x + td_x) + b(o_y + td_y) + c(o_z + td_z) + d = 0$$

$$t = \frac{-(ao_x + bo_y + co_z + d)}{ad_x + bd_y + cd_z}$$

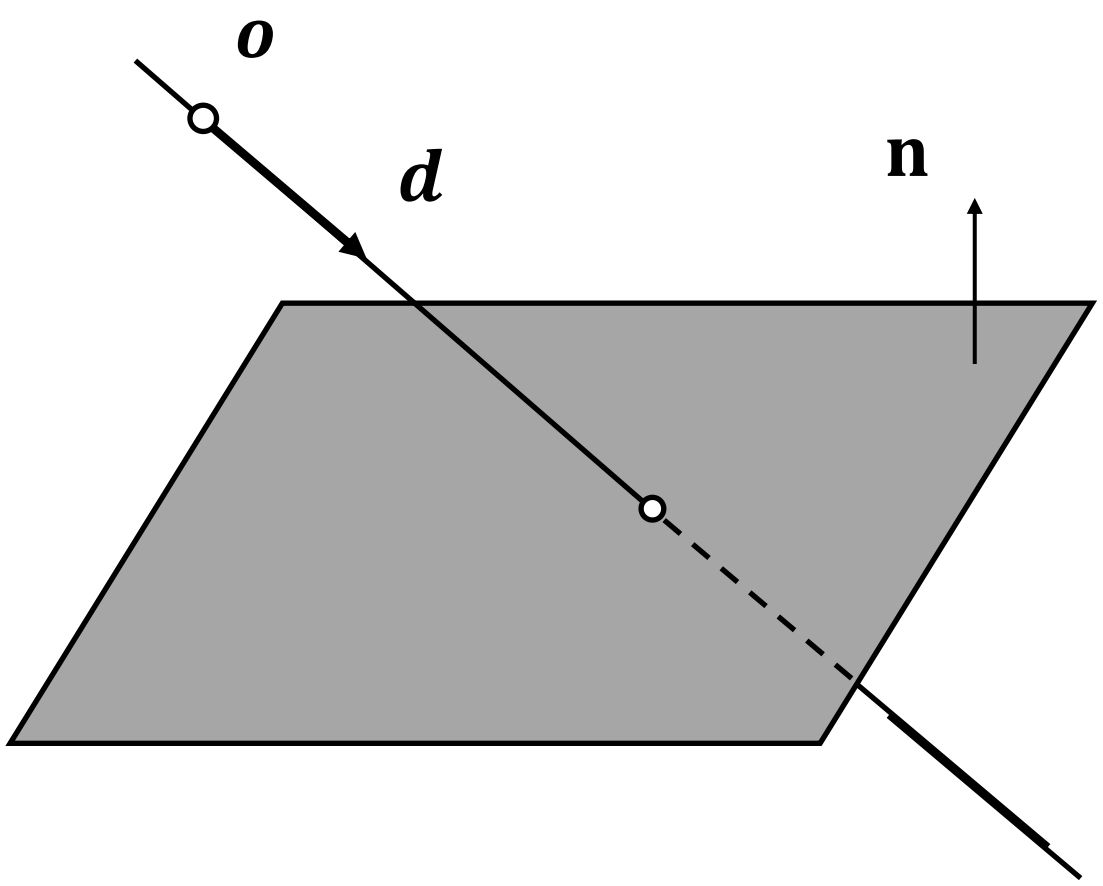
$$t = \frac{-(\mathbf{n} \cdot \mathbf{o} + d)}{\mathbf{n} \cdot \mathbf{d}}$$



相交测试

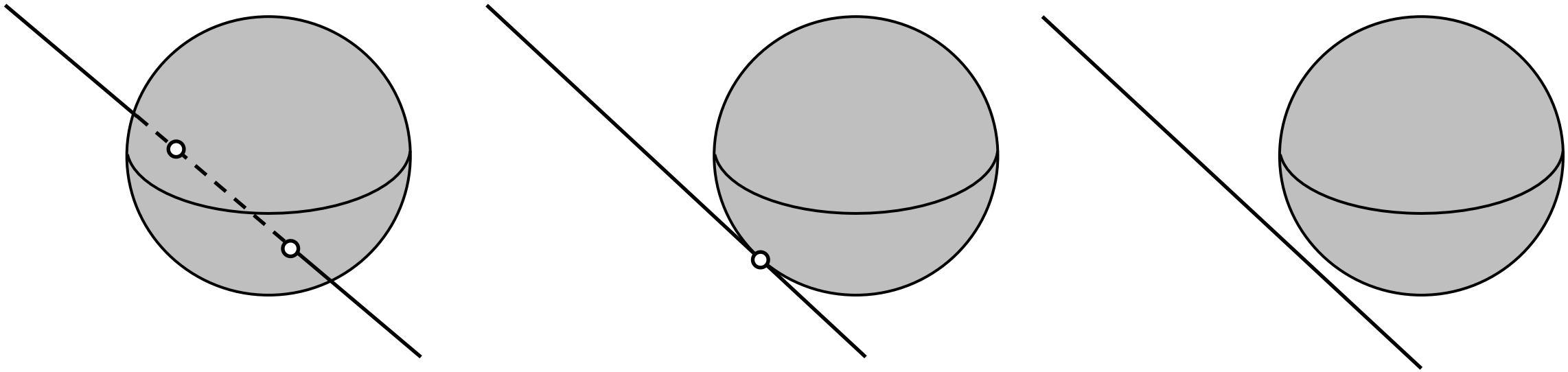
- 线性结构与平面的相交

```
template<typename Real>
int  TLine3D<Real>::intersect(const
        TPlane3D<Real>& plane,
        TPoint3D<Real>& interPt) const
{
        Real DdN = direction.dot(plane.normal);
        if (abs(DdN) < REAL_EPSILON)
        {
                return 0;
        }

        Coord3D offset = origin - plane.origin;
        Real t = -offset.dot(plane.normal) / DdN;

        interPt.origin = origin + t * direction;
        return 1;
}
```



相交测试

- 线性结构与球的相交



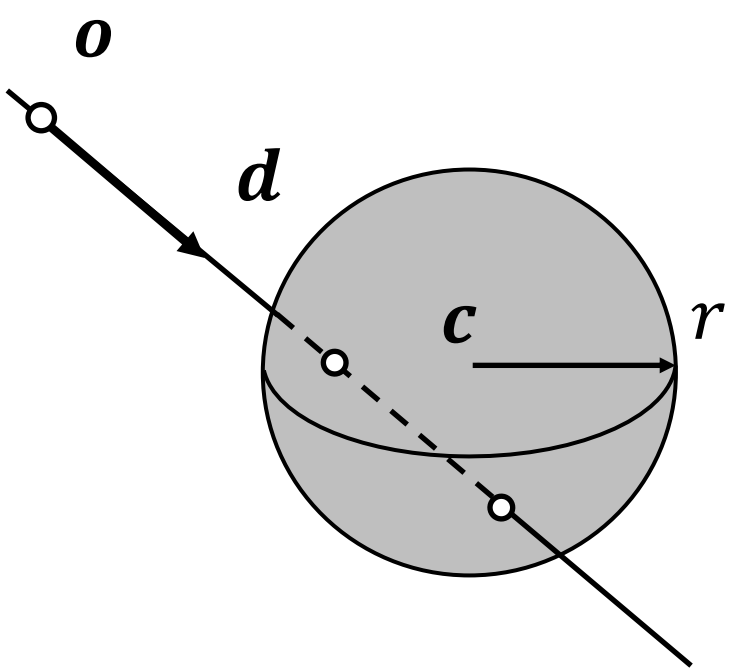
Two intersections　　One intersection　　No intersection

相交测试

- 线性结构与球的相交

直线：$\mathbf{x}(t) = \mathbf{o} + t\mathbf{d}, t \in (-\infty, \infty)$

球面：$\|\mathbf{x} - \mathbf{c}\|^2 = r^2$



$$\|\mathbf{o} + t\mathbf{d} - \mathbf{c}\|^2 = r^2$$

$$(t\mathbf{d} + \mathbf{o} - \mathbf{c}) \cdot (t\mathbf{d} + \mathbf{o} - \mathbf{c}) = r^2$$

$$t^2(\mathbf{d} \cdot \mathbf{d}) + 2t\mathbf{d} \cdot (\mathbf{o} - \mathbf{c}) + (\mathbf{o} - \mathbf{c}) \cdot (\mathbf{o} - \mathbf{c}) - r^2 = 0$$

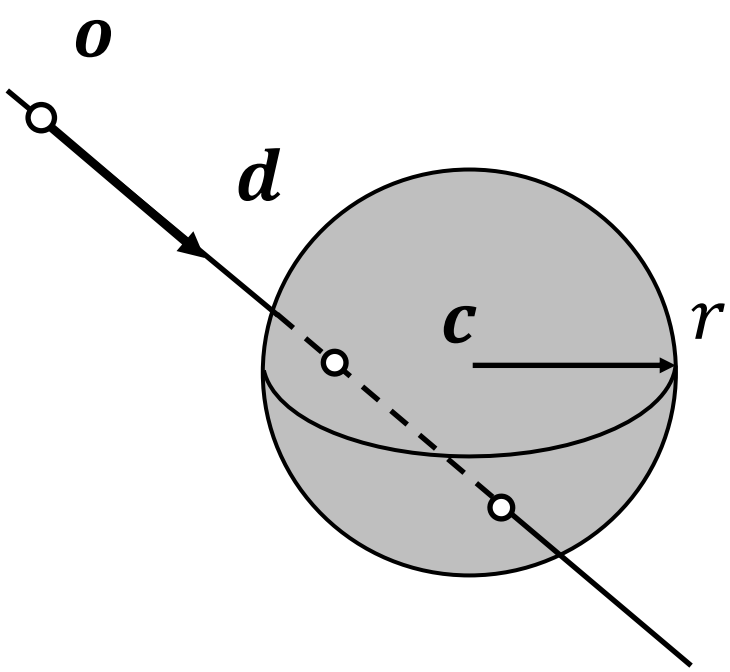
$$t^2 + 2t\mathbf{d} \cdot (\mathbf{o} - \mathbf{c}) + (\mathbf{o} - \mathbf{c}) \cdot (\mathbf{o} - \mathbf{c}) - r^2 = 0 \quad \Longleftrightarrow \quad at^2 + 2bt + c = 0$$

相交测试

- 线性结构与球的相交

直线：$\mathbf{x}(t) = \mathbf{o} + t\mathbf{d}, t \in (-\infty, \infty)$

球面：$\|\mathbf{x} - \mathbf{c}\|^2 = r^2$



$$at^2 + 2bt + c = 0$$

$$a = 1$$

$$b = t\mathbf{d} \cdot (\mathbf{o} - \mathbf{c})$$

$$c = (\mathbf{o} - \mathbf{c}) \cdot (\mathbf{o} - \mathbf{c}) - r^2$$

$$t_0 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}, t_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

Two intersections	$b^2 - 4ac > 0$
One intersection	$b^2 - 4ac = 0$
No intersection	$b^2 - 4ac < 0$

相交测试

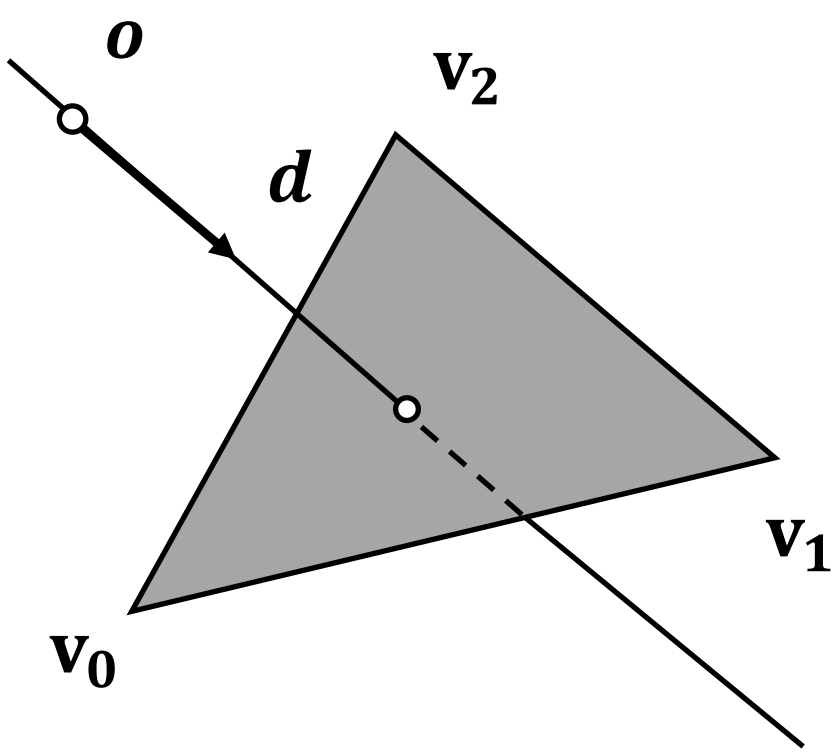
- 线性结构与三角形的相交

直线：$\mathbf{x}(t) = \mathbf{o} + t\mathbf{d}, t \in (-\infty, \infty)$

三角形：$\mathbf{x}(u, v, w) = w\mathbf{v}_0 + u\mathbf{v}_1 + v\mathbf{v}_2, u + v + w = 1$

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

$$\begin{bmatrix} -\mathbf{d} & \mathbf{v}_1 - \mathbf{v}_0 & \mathbf{v}_2 - \mathbf{v}_0 \end{bmatrix} \begin{bmatrix} t \\ u \\ v \end{bmatrix} = \begin{bmatrix} \mathbf{o} - \mathbf{v}_0 \end{bmatrix} \Leftrightarrow \mathbf{Ax} = \mathbf{b}$$

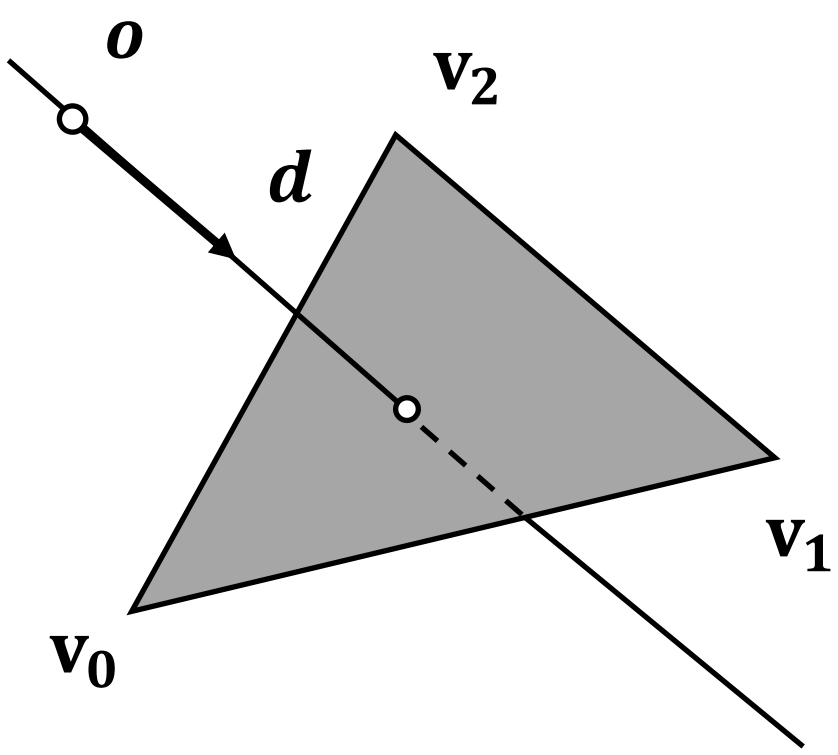


相交测试

- 线性结构与三角形的相交

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{D}\begin{bmatrix} \left(\left(\mathbf{o}-\mathbf{v}_0\right)\times\left(\mathbf{v}_1-\mathbf{v}_0\right)\right)\cdot\left(\mathbf{v}_2-\mathbf{v}_0\right) \\ \left(\mathbf{d}\times\left(\mathbf{v}_2-\mathbf{v}_0\right)\right)\cdot\left(\mathbf{o}-\mathbf{v}_0\right) \\ \left(\left(\mathbf{o}-\mathbf{v}_0\right)\times\left(\mathbf{v}_1-\mathbf{v}_0\right)\right)\cdot\mathbf{d} \end{bmatrix}$$

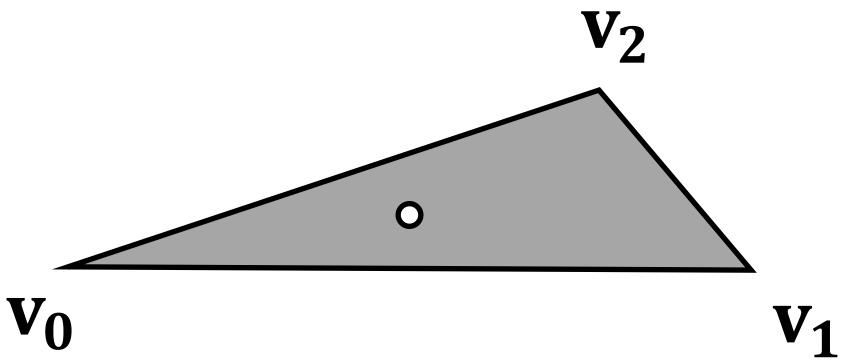
$$D = \mathbf{d}\times\left(\mathbf{v}_2-\mathbf{v}_0\right)\cdot\left(\mathbf{v}_1-\mathbf{v}_0\right)$$



相交测试

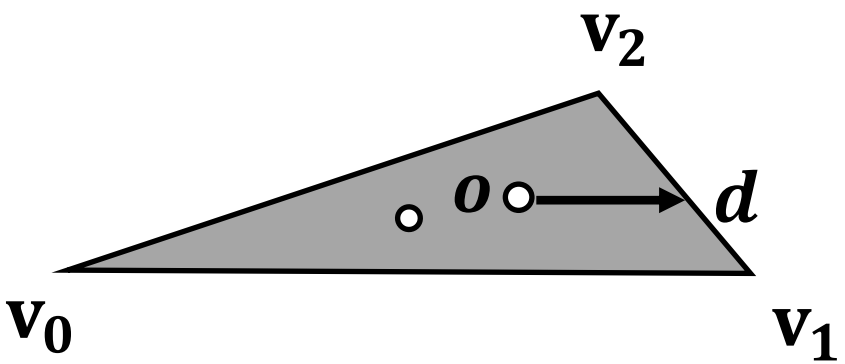
- 线性结构与三角形的相交

Case 1:



$$D = \mathbf{d} \times \left(\mathbf{v}_2 - \mathbf{v}_0\right) \cdot \left(\mathbf{v}_1 - \mathbf{v}_0\right) = 0$$

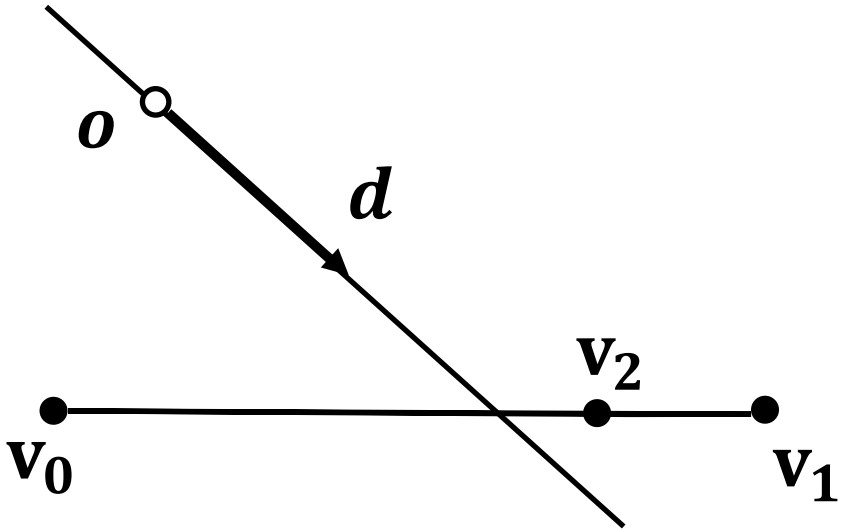
Case 2:



相交测试

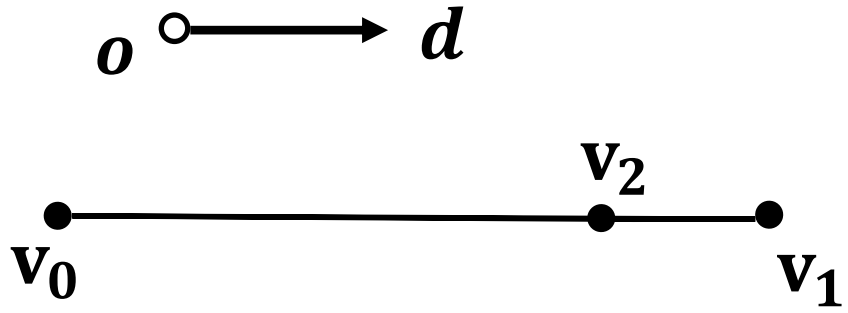
- 线性结构与四面体的相交

Case 3:



$$D = \mathbf{d} \times \left(\mathbf{v}_2 - \mathbf{v}_0\right) \cdot \left(\mathbf{v}_1 - \mathbf{v}_0\right) = 0$$

Case 4:



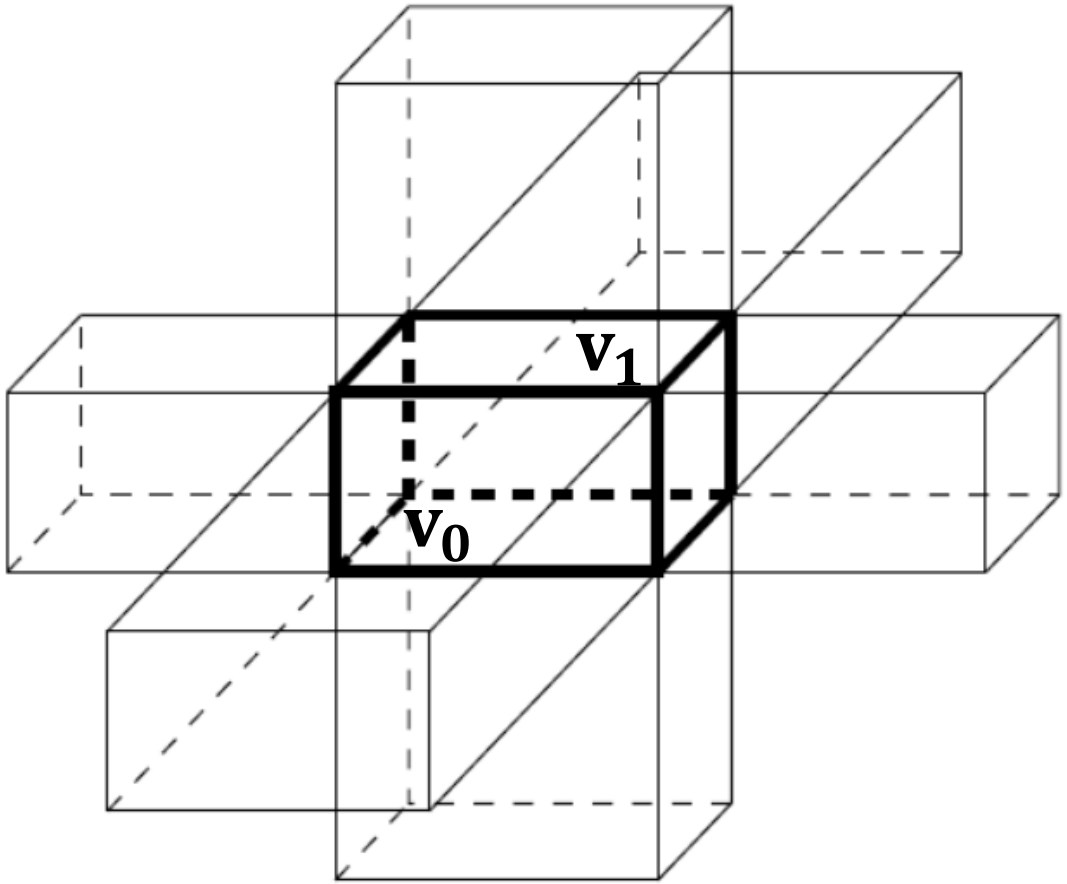
相交测试

- 线性结构与AABB的相交

Slab x: $[x_0, x_1]$

Slab y: $[y_0, y_1]$

Slab z: $[z_0, z_1]$



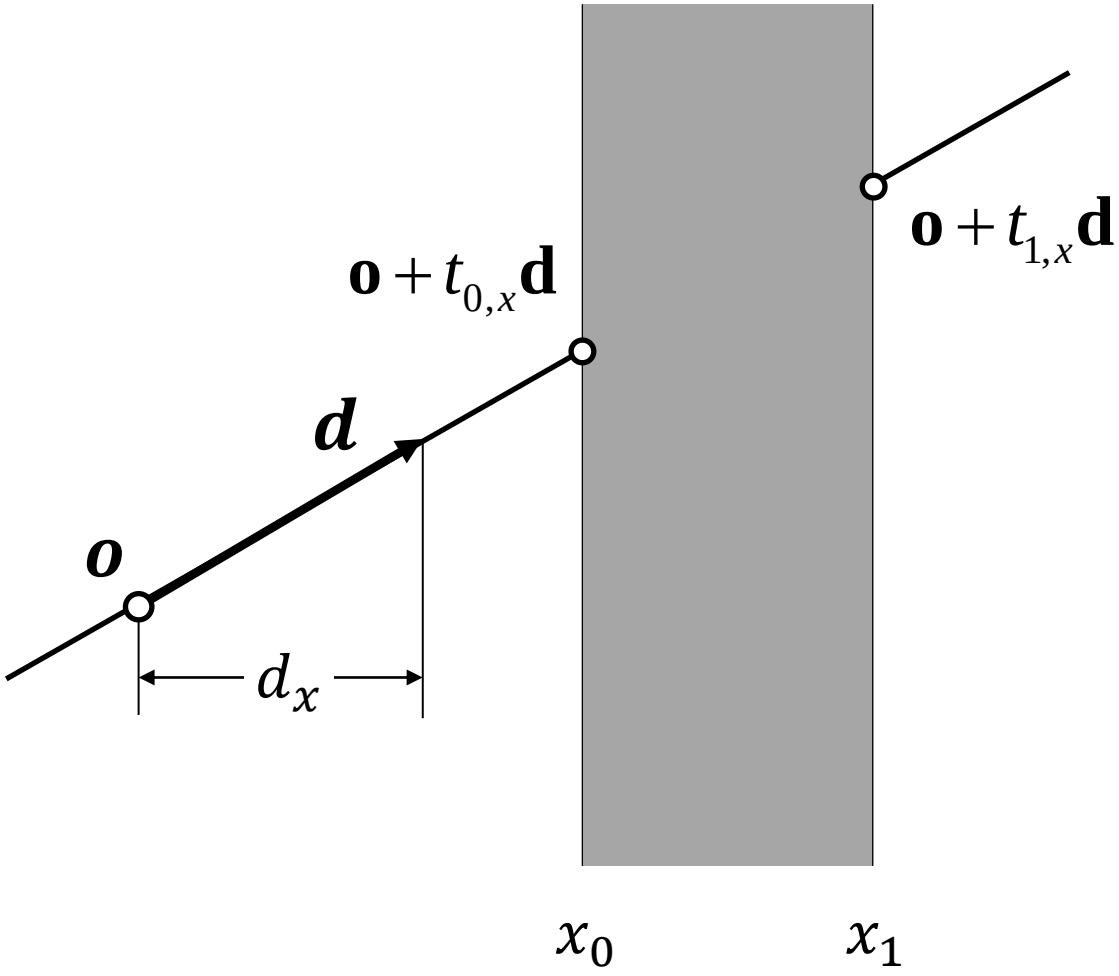
Slabs method [Kay and Kajiya (1986)]

相交测试

- 线性结构与AABB的相交

$$t_{0,x} = \frac{x_0 - o_x}{d_x}$$

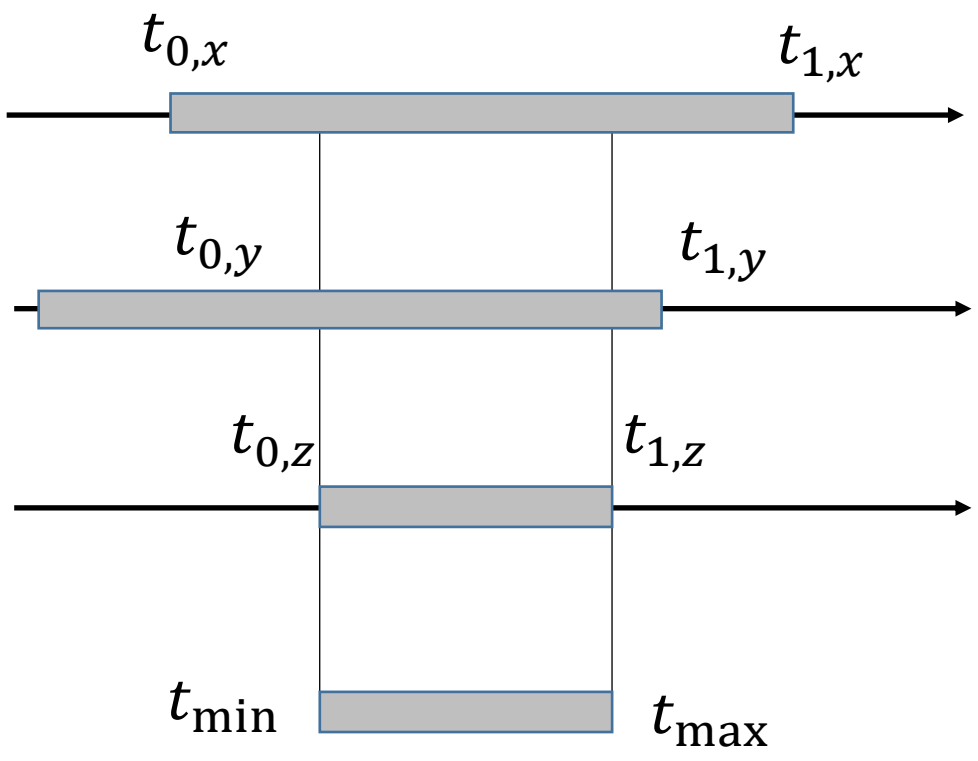
$$t_{1,x} = \frac{x_1 - o_x}{d_x}$$

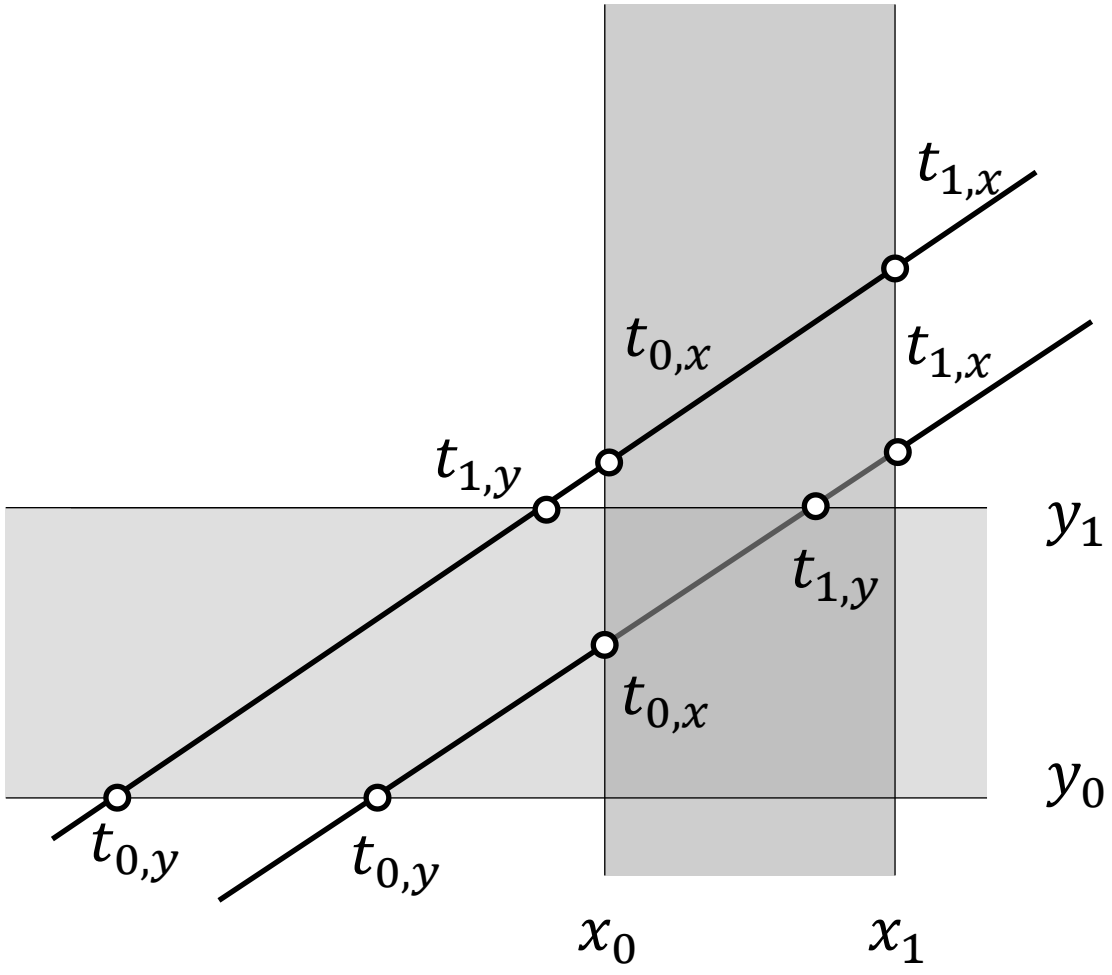


Slabs method [Kay and Kajiya (1986)]

相交测试

- 线性结构与AABB的相交



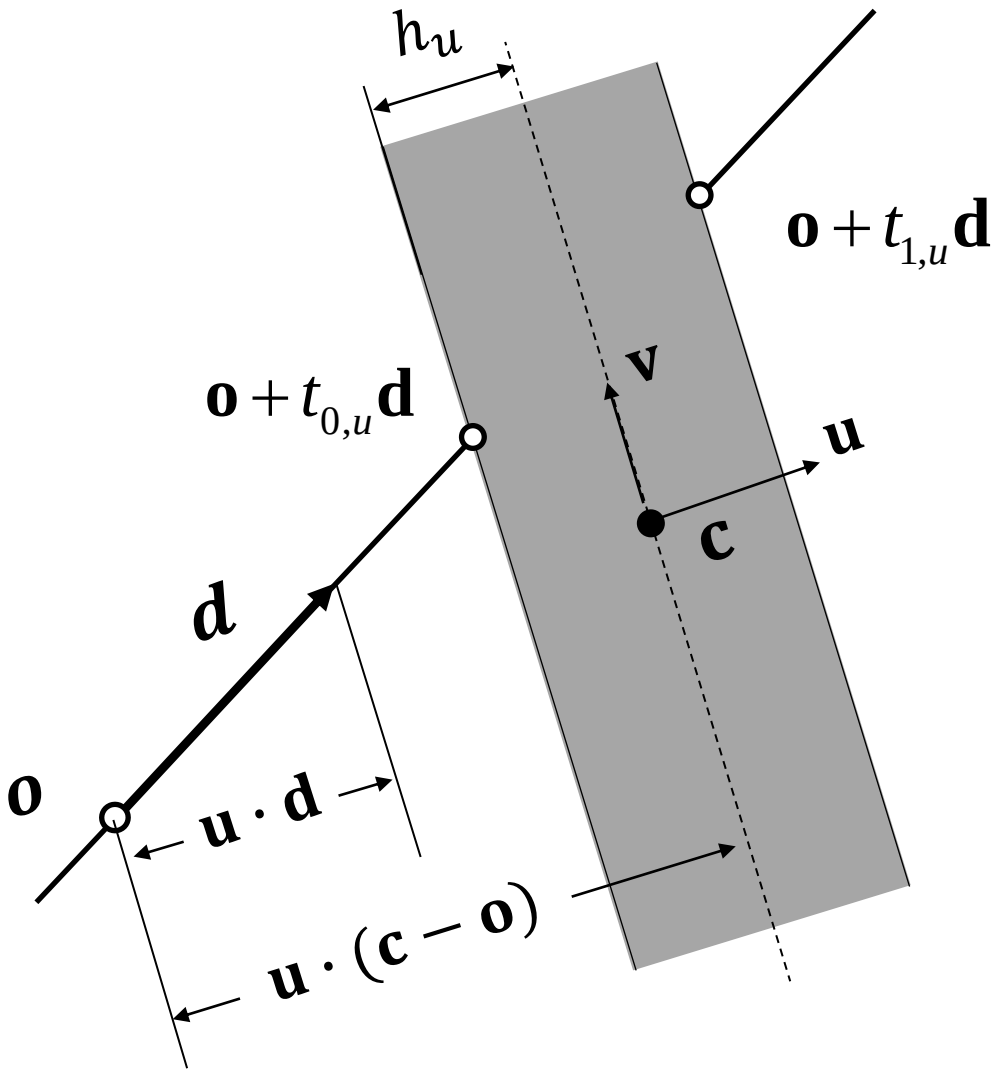


相交测试

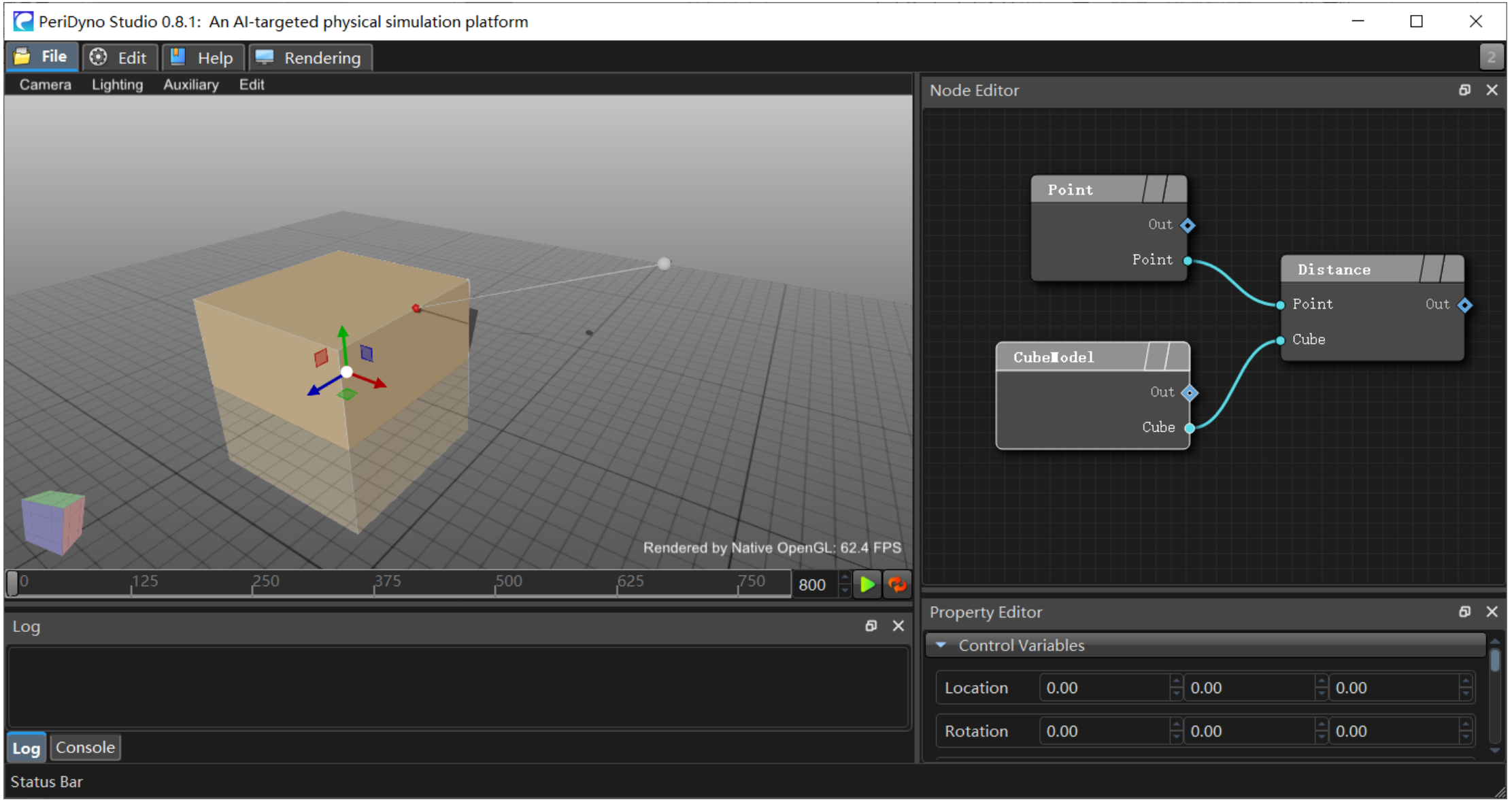
- 线性结构与OBB的相交

$$t_{0,u} = \frac{\mathbf{u}\cdot(\mathbf{c}-\mathbf{o}) - h_u}{\mathbf{u}\cdot\mathbf{d}}$$

$$t_{1,u} = \frac{\mathbf{u}\cdot(\mathbf{c}-\mathbf{o}) + h_u}{\mathbf{u}\cdot\mathbf{d}}$$

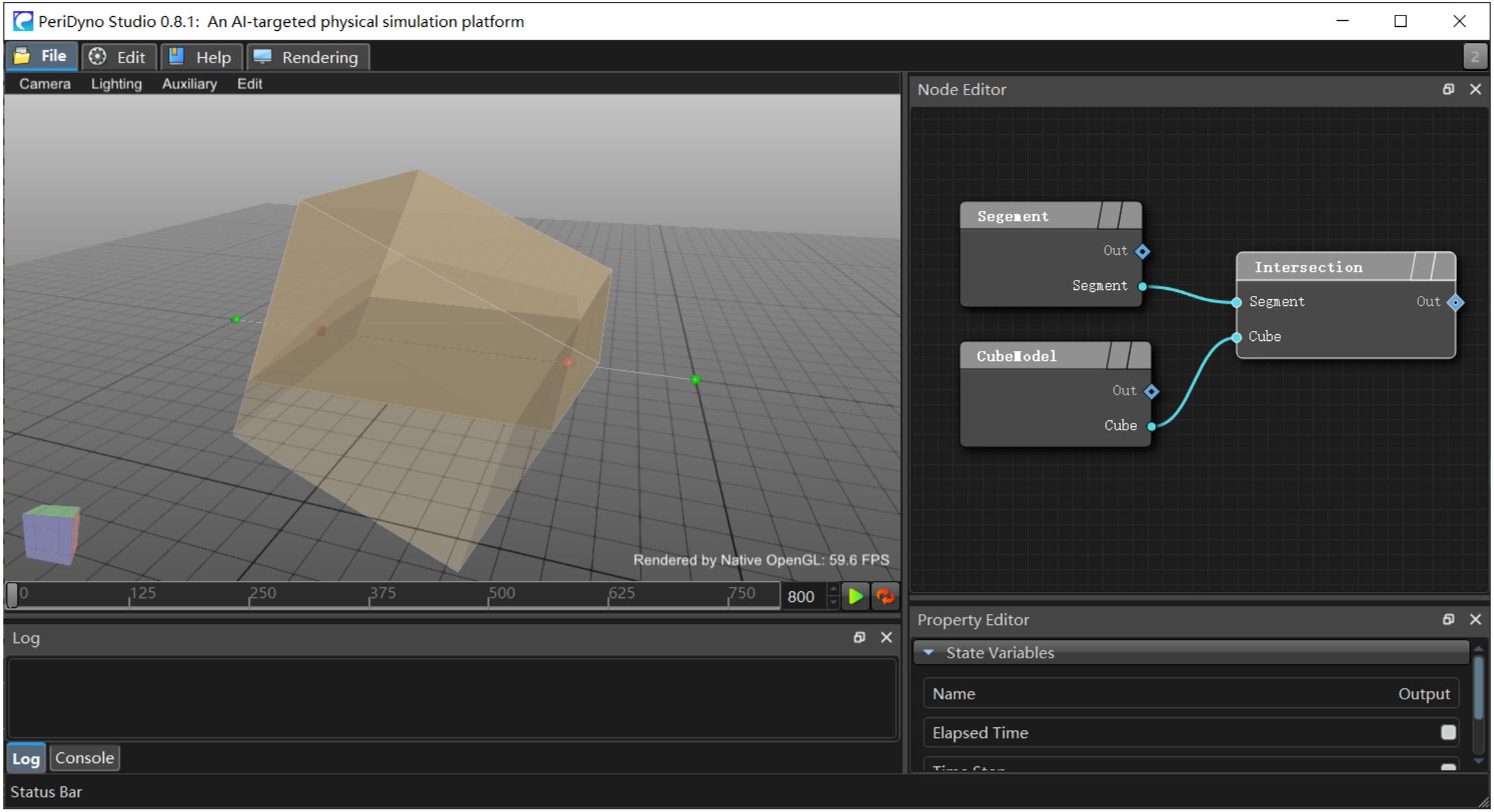


测试场景一：点与OBB距离计算



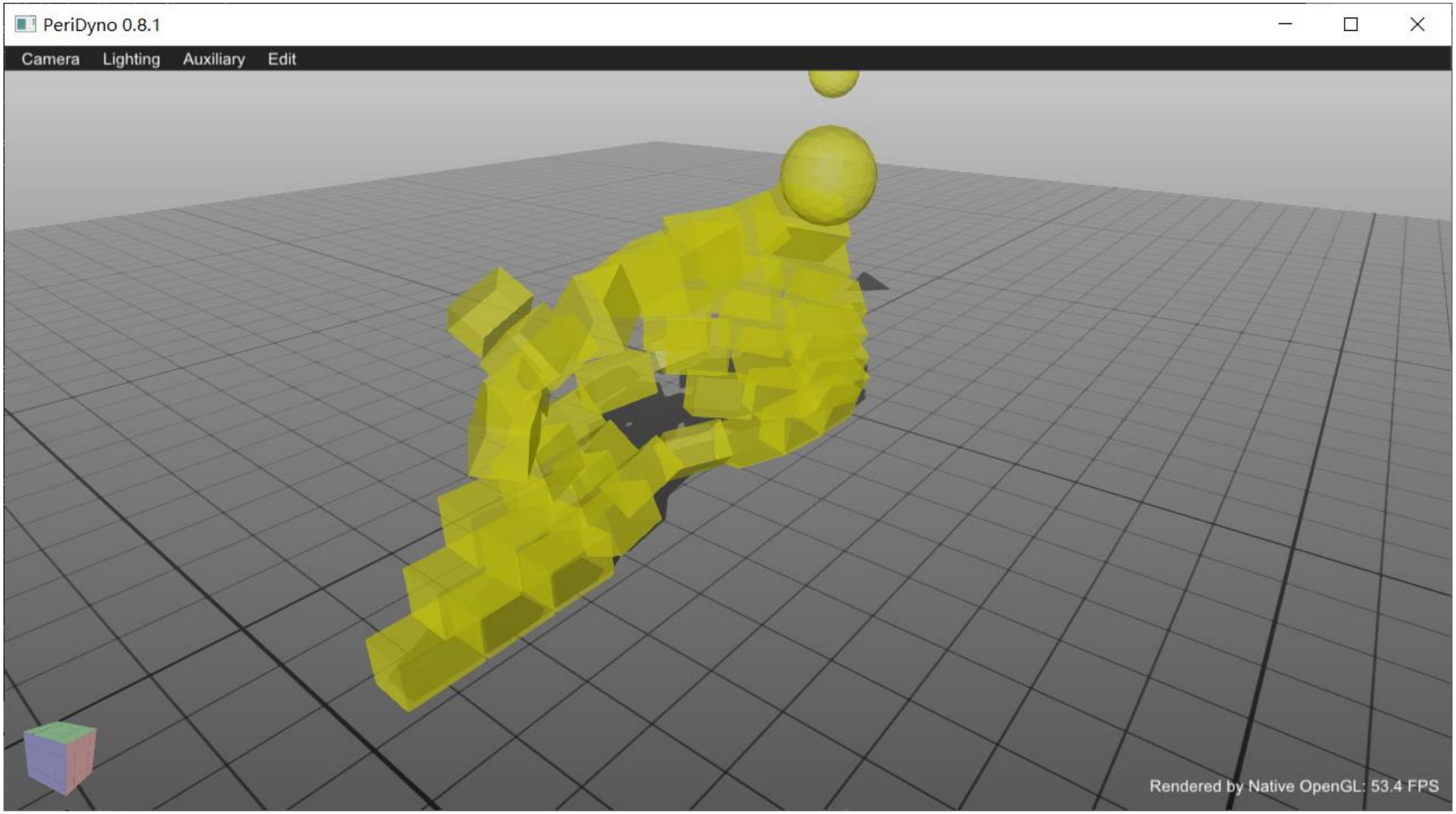
example/Cuda/RigidBody/ Qt_ComputeDistance

测试场景二：线段与OBB相交测试



example/Cuda/RigidBody/ Qt_IntersectionBetweenLineAndOBB

测试场景三：刚体动力学



example/Cuda/RigidBody/GL_HitBricks

参考资料

- Geometric tools for computer graphics (Elsevier, 2003)

Question