Contents

[碰撞检测 1](#_Toc32954417)

[AABB碰撞检测 1](#_Toc32954418)

[两线相交碰撞检测 1](#_Toc32954419)

[线与矩形碰撞检测 2](#_Toc32954420)

[圆与AABB碰撞检测 2](#_Toc32954421)

[几何测量 3](#_Toc32954422)

[直线之间最短距离 3](#_Toc32954423)

# 碰撞检测

## AABB碰撞检测

bool AABBCheck(Rectangle P1, Rectangle P2) {

if(P1.x < P2.x + P2.width &&

P1.x + P1.width > P2.x && P1.y < P2.y + P2.height && P1.y + P1.height > P2.y)

return true

else return false

}

## 两线相交碰撞检测

基本就是两条线段参数方程的求解过程

bool LineLineCheck(Line L1, Line L2) {

L1 (x1, y1) to (x2, y2)

L2 (x3, y3) to (x4, y4)

t1 = ((x4 - x3)\*(y1 - y3) - (y4 - y3)\*(x1 - x3)) / ((y4 - y3)\*(x2 - x1) - (x4 - x3)\*(y2 - y1));

t2 = ((x2 - x1)\*(y1 - y3) - (y2 - y1)\*(x1 - x3)) / ((y4 - y3)\*(x2 - x1) - (x4 - x3)\*(y2 - y1));

if (t1 >= 0 && t1 <= 1 && t2 >= 0 && t2 <= 1)

return true

else return false

}

## 线与矩形碰撞检测

bool LineRectangleCheck(Line L, Rectangle R) {

4 times LineLineCheck(L, L1-4 of R)

if (any LineLineCheck() is true)

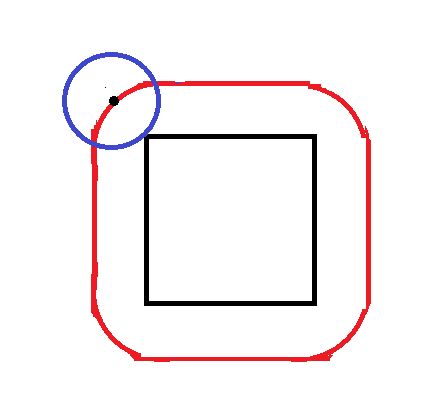
return true

else return false

}

## 圆与AABB碰撞检测

先计算圆和AABB之间的最短距离，看是否小于圆半径



int CircleAABBsquaredDistance(vec2 circleCentre, int circleRadius, vec2 boxStart, boxWidth, boxHeight) {

float dx = std::max(boxStart.x - circleCentre.x, 0.0f);

dx = std::max(dx, circleCentre.x - (boxStart.x + boxWidth));

float dy = std::max(boxStart.y - circleCentre.y, 0.0f);

dy = std::max(dy, circleCentre.y - (boxStart.y + boxHeight));

return (dx \* dx) + (dy \* dy);

} //此计算方法非常巧妙，将多种情况下的计算合并为简练的判断式

bool CircleAABBCheck(circle, rectangle) {

if(circleAABBsquaredDistance(…) <= (circleRadius \* circleRadius))

return true

else return false

}

# 几何测量

## 直线之间最短距离

minSquaredDistanceLineLine(Line Line1, Line Line2) { //两点式或点向式

vec3 u = line1End - line1Start;

vec3 v = line2End - line2Start;

vec3 w = line1Start - line2Start;

float a = Util::dot(u, u); // always >= 0

float b = Util::dot(u, v);

float c = Util::dot(v, v); // always >= 0

float d = Util::dot(u, w);

float e = Util::dot(v, w);

float D = a \* c - b \* b; // always >= 0

float sc, tc; //参数方程参数

// compute the line parameters of the two closest points

if (D < Util::EPSILON) { // the lines are almost parallel

sc = 0.0;

tc = (b > c ? d / b : e / c); // use the largest denominator

}

else { //计算空间上最接近的位置

sc = (b\*e - c \* d) / D;

tc = (a\*e - b \* d) / D;

}

// get the difference of the two closest points

vec3 dP = w + (sc \* u) - (tc \* v); // = L1(sc) - L2(tc)

float norm = Util::dot(dP, dP);

return norm;

}