isScreenPointInRect

不管该Node做何种几何变化，都是可以正常区域判断的，比如旋转，放缩等（已证明）

**bool isScreenPointInRect(const Vec2 &pt, const Camera\* camera, const Mat4& w2l, const Rect& rect, Vec3 \*p)**

{

if (nullptr == camera || rect.size.width <= 0 || rect.size.height <= 0)

{

return false;

}

// first, convert pt to near/far plane, get Pn and Pf

Vec3 Pn(pt.x, pt.y, -1), Pf(pt.x, pt.y, 1);

Pn = camera->unprojectGL(Pn);

Pf = camera->unprojectGL(Pf);

// then convert Pn and Pf to node space

w2l.transformPoint(&Pn);

w2l.transformPoint(&Pf);

// Pn and Pf define a line Q(t) = D + t \* E which D = Pn

auto E = Pf - Pn;

// second, get three points which define content plane

// these points define a plane P(u, w) = A + uB + wC

Vec3 A = Vec3(rect.origin.x, rect.origin.y, 0);

Vec3 B(rect.origin.x + rect.size.width, rect.origin.y, 0);

Vec3 C(rect.origin.x, rect.origin.y + rect.size.height, 0);

B = B - A;

C = C - A;

// the line Q(t) intercept with plane P(u, w)

// calculate the intercept point P = Q(t)

// (BxC).A - (BxC).D

// t = -----------------

// (BxC).E

Vec3 BxC;

Vec3::cross(B, C, &BxC);

auto BxCdotE = BxC.dot(E);

if (BxCdotE == 0) {

return false;

}

auto t = (BxC.dot(A) - BxC.dot(Pn)) / BxCdotE;

Vec3 P = Pn + t \* E;

if (p) {

\*p = P;

}

return rect.containsPoint(Vec2(P.x, P.y));

}