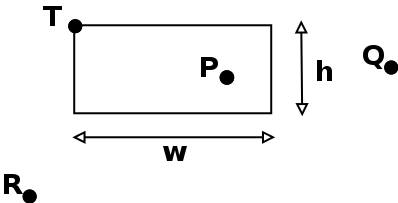
## Math for COMP258

**Collision Detection**

*Does a point lie inside a rectangle with top left corner point* T*, width* w *and height* h*?*



boolean ptInRect(PVector point,PVector T,float w,float h) {

boolean result = false;

if (point.x >= T.x && point.x <= T.x + w)

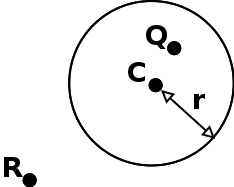
if (point.y >= T.y && point.y <= T.y + h)

result=true;

return result;

}

*Does a point lie inside a circle with center* C*, and radius* r*?*



boolean ptInCircle(PVector point,PVector C,float r) {

boolean result = false;

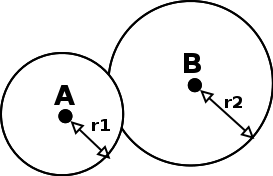
if (point.dist(C)<=r)

result=true;

return result;

}

*Do two circles with centers* A *and* B*, and radii* r1 *and* r2*, intersect?*



boolean circleCollision(PVector A, PVector B,float r1, float r2) {

boolean result = false;

if (A.dist(B)<=r1+r2)

result=true;

return result;

}

**Points around the Perimeter of a Circle**

See <http://www.mathopenref.com/coordparamcircle.html> for a nice, interactive demonstration

PVector perimeterPt(PVector center, float radius, float angle) {

PVector result = new PVector();

result.x = center.x + radius \* cos(angle);

result.y = center.y + radius \* sin(angle);

return result;

}

Note that in Processing the default measurement for angles is **radians** which range from 0 to TWO\_PI. **Degrees**, on the other hand, range from 0 to 360. Processing provides a useful function called **radians()** to convert degrees to radians.

For example, to calculate the point at 60 degrees around a circle centered on (25,30) with radius 10:

PVector center = new PVector(25,30);

PVector pt = perimeterPt(center, 10, radians(60));

**Gaussian vs Uniform Random Distribution**

**Basic Physics**