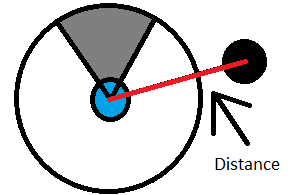
**Player Detection**

**Ru Li**

Overview:

- Determine if player has been detected by a guard.

- Blue dot is the Guard

- Black dot is the player

- Circle surrounding guard is the vision radius of the guard

- Grey arc is the detection arc (If player walking inside it, player is detected)

1. Find if player is in guard's vision circle.

Let d represent the distance between the center of the player circle and the center of the guard's circle.

d = sqrt( (player.x - guard.x)^2 + (player.y - guard.y)^2)

Using Pythagorean Theorem we can find the distance.

Total radius = player.radius + guard.radius

We will need the total radius to determine if player circle is touching the guard's circle

If d > total radius:

player is not in range of guard's circle. Don't need to run the detection function.

If d = total radius:

player circle is touching guard's circle at exactly one point. Run detection function.

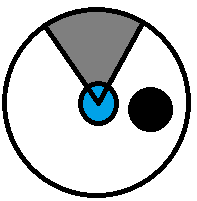
if d < total raidus:

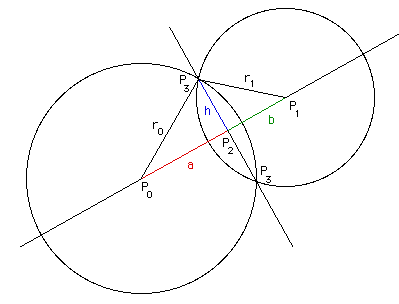
player is either intersecting the guard's circle at a maximum of two points or is inside the guard's circle and not intersecting at all. Since there's not guaranteed to be an intersection, we don't know if we need to run the detection function or not. This problem will be solved by using layers

which will be explained later on.

2. If player is inside or touching the guard's circle, find where it's intersecting.

- Although player is in guard's vision range, that doesn't mean guard is looking at player

To find if guard is looking at player, intersection points between the player circle and the guard circles are needed.

a2 + h2 = r02

and

b2 + h2 = r12

d = a + b

Using this information, we can solve for a

1. a2 - r12 = -h2

2. (d-a)2 - r02 = -h2

Combine equations

-> a2 - r12 = (d-a)2 - r02

-> a2 - r12 - (d-a)2 = - r02

-> a2 - 2da - d2 - a2 = - r02 - r12

Isolate a

-> a = (d2 - r12 - r02)/(2d)

Solve h by substituting a into a2 + h2 = r02

h2 = r02 - (a)2

P3 = (x3,y3) and P2 = (x2,y2) and P1 = (x1,y1) and P0 = (x0,y0)

x2 = x0 + a(x1 - x0)/d

y2 = y0 + a(y1 - y0)/d

x3 = x2 + h(y1 - y0)/d

x4 =x2 - h(y1 - y0)/d

y3 = y2 + h(x1 - x0)/d

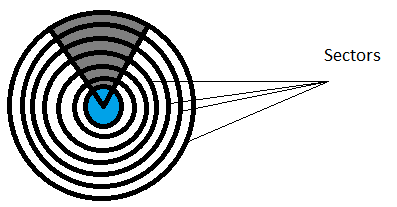
y4 = y2 - h(x1 - x0)/d

(x3,y3) are the intersection points

3. What if player is complete inside the guard's vision range?

- There's no intersection of player is inside the guard's vision range

Solution: LAYERS

- Round distance (d value) to the nearest 20 and use that sector to get intersection points

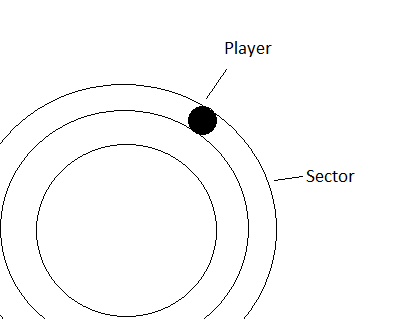
- Player is 20 units wide

- Sectors are 20 units apart

- If player is inside the guard's circle, player must touch at least one sector.

- Sector acts as the radius

- Sectors allows us to find intersections no matter where the player is inside the guard's circle



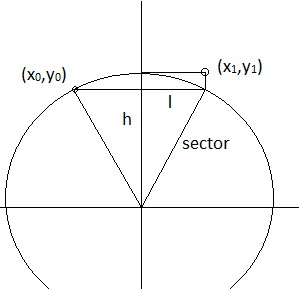
If d <= total radius

sector =int(20\*round(float(d)/20))

- Instead of using the radius of guard's circle, use sector

- Rounding sector to nearest multiple of 20 is faster than tracking all of the possible sectors

4. Finding player in the guard's vision range.

- Now that we know exactly where the player is, we need to find when player is in the guard's arc

- All guard's vision arcs are 80 degrees

So

h = sector\*cos(40)

l = sector\*sin(40)

detect\_x0 = guard.x - l

detect\_y0 = guard.y - h

detect\_x1 = guard.x + l

detect\_y1 = guard.y - sector

Using these measurements

if detect\_x0 < x3 < detect\_x1

or detect\_x0 < x4 < detect\_x1:

if detect\_y0 > y3 > detect\_y1

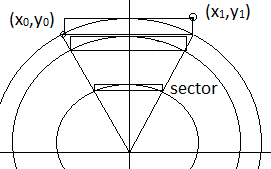
or detect\_y0 > y4 > detect\_y1:

detection = True

So if the player circle-guard circle intersection is inside of the rectangle, it means that the player has been detected inside the guard's vision arc.

Measurements above are for when the guard is looking up. Different equations are needed for when the guard looks left, down, right and when the guard is scanning an area.

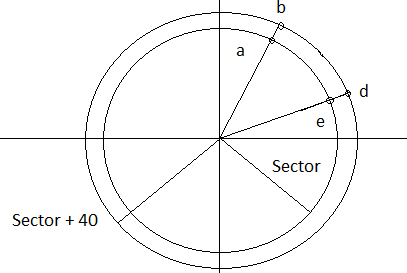
The sector adjusts the code above so that the rectangle zone of detection will always be in the right place.



5. Finding player in the guard's vision range while the guard is scanning an area.

- Sometimes the guard would scan the area by having the start of his arc from a degrees to b degrees

For example: guard scans from 80 degrees to 260 degrees

For this problem we add another radius that's sector+40.

The strategy from the previous problem (making a detection rectangle) won't work. We don't have easy measurements like the last problem where the only 4 options for the arc was up, down, left or right. In this problem, the arc can have any angle.

The 5 mini circles (a, b, d and e) are all points that we can get the measure of.

The line from b to d must either not touch the sector at all or touch the sector circle as a tangent.

Basically, we find the x's and the y's that will give us the rectangle that we use to test if player is inside the guard's vision or not. Using these 4 points, we can find the rectangle in all 4 quadrants.

arc\_x0 = min ( a, b)

arc\_x0 = a

arc\_x1 = min (e, d)

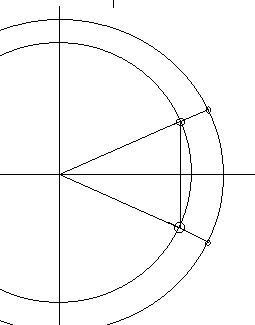
arc\_x1 = e

arc\_y0 = max (a, b)

arc\_y0 = a

arc\_y1 = max(e, d)

These equations only work for the first quadrant. Max and min will need to be adjusted for the other quadrants.

Here's an example of where the new way works better.

arc\_x0 = max (a, d)

arc\_x1 = min (b, e)

arc\_y0 = max (a, b)

arc\_y1 = min (d, e)

We can use the equation in the previous problem for this one because the angle start is 40 but if it was 41 or 39, we would have to use this method.