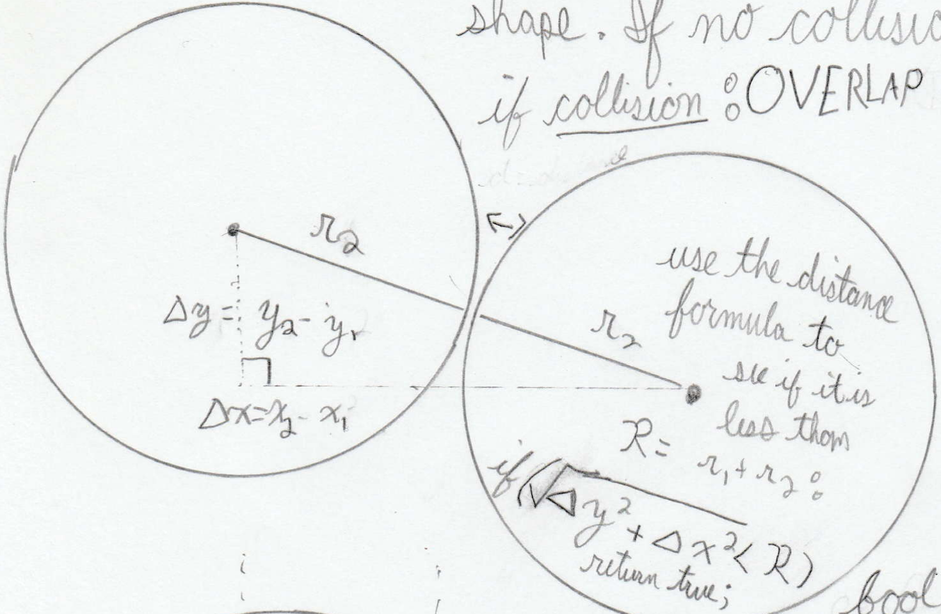
Method 3: First, calculate a circular perimeter around each shape. If no collision return NO_OVERLAP. if <u>collision</u>: OVERLAP

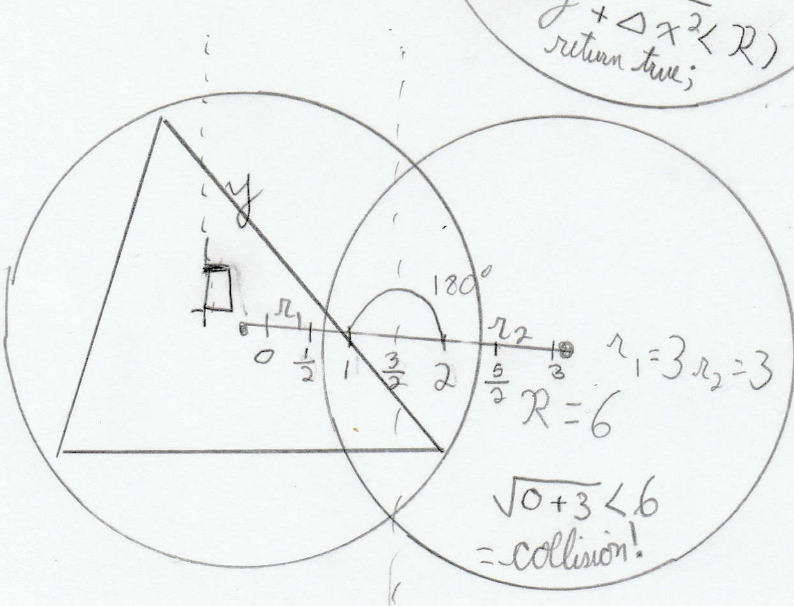


I. If Shape1.type==circle and Shape2.type==circle, return true

II. If the shapes were not circles, pass each polygon to the function:

bool Polygon::overlaps(const Polygon&);

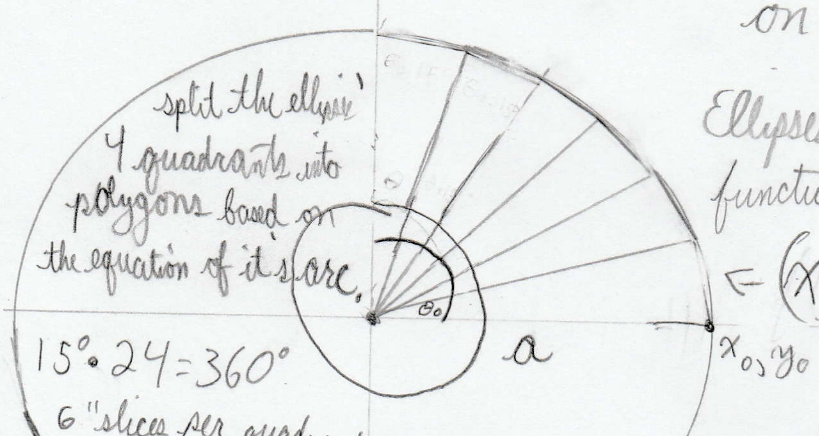
this function implements the open-source API known a Clipper. By integrating this free-for-commercial-use library, the binary will now be able to detect clipping (collision) on *any* two polygons!



Ellipses will need to be converted by my function, function polygonFromEllipse(Ellipse&);

⇐ $(x_i, y_i) = a\cos(15° \cdot i), b\sin(15° \cdot i)$

so we increment by 15° to draw a 24-sided polygon!


split the ellipse's 4 quadrants into polygons based on the equation of it's arc.
θ_0
a
x_0, y_0
15°·24=360°
6 "slices per quadrant, now we have coordinates! 24 of them!
b

Now, we can detect collision of all shapes!