1.) The position vector is $\bar{p}(t) = [a\cos(2\pi t),\ b\sin(2\pi t)]^T$.

The tangent is $\frac{\partial \bar{p}}{\partial t} = [-2\pi a\sin(2\pi t),\ 2\pi b\cos(2\pi t)]^T$.

Since we are working in 2D, we can use the fact that any vector $[p, q]^T$ has $[-q, p]$ as a perpendicular vector.

So ~~the~~ a normal vector is $[-2\pi b\cos(2\pi t),\ -2\pi a\sin(2\pi t)]^T$.

Notice the given ellipse satisfies the implicit equation

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1.$$

A circle of radius 1 ~~satisfy~~ satisfies $x_c^2 + y_c^2 = 1$. So, by inspection, we want a transformation matrix that ~~makes~~ takes $x \to \frac{x}{a}$, $y \to \frac{y}{b}$. This matris is $\begin{bmatrix} \frac{1}{a} & 0 & 0 \\ 0 & 1/b & 0 \\ 0 & 0 & 1 \end{bmatrix}$ (homogeneous coordinates).

2.) At the intersection point $\bar{q}$ both the circle and line equations are satisfied. Thus we have $\bar{q} = \bar{p}_0 + \lambda_q \vec{d}$ and $\|\bar{q} - \bar{p}_1\|^2 = r^2$.

Substituting the line equation into the circle equation gives

$$\|\bar{p}_0 + \lambda_q \vec{d} - \bar{p}_1\|^2 = r^2.$$

Expanding, after some algebra, we obtain

$$\|\vec{d}\|^2 \lambda_q^2 + 2\lambda_q (\bar{p}_0 - \bar{p}_1)\cdot\vec{d} + \|\bar{p}_0 - \bar{p}_1\|^2 - r^2 = 0$$

Notice this is a quadratic eq'n. in λ_q with

$$A\lambda_q^2 + B\lambda + C = 0 \qquad A = \|\vec{d}\|^2 \quad B = 2(\bar{p}_0 - \bar{p}_1)\cdot\vec{d} \quad C = \|p_0 - p_1\|^2 - r^2$$

By considering the discriminant $D = B^2 - 4AC$, we can determine the number of solutions and their locations.

If $D<0$ no real solutions for λ_q $\rightarrow$ no intersections.

$D=0$ 1 real solution $\lambda_q = \frac{-B}{2A}$

$D>0$ 2 real solutions $\lambda_q = \frac{-B \pm \sqrt{D}}{2A}$.

If there is a real solution for λ_q, we can obtain the intersection point location using the line eq'n $\bar{q} = \bar{p}_0 + \lambda_q \vec{d}$.

Algorithm

1. Compute A, B, & C from the above expressions.
2. Compute the discriminant D.
3. if $D<0$ return "no intersections"

   if $D=0$ $\lambda = \frac{-B}{2A}$ $\bar{q} = \bar{p}_0 + \lambda \vec{d}$

   return "one intersection @ $\bar{q}$"

   if $D>0$

   $\lambda_1 = \frac{-B+\sqrt{D}}{2A}$ $\lambda_2 = \frac{-B-\sqrt{D}}{2A}$

   $\bar{q}_1 = \bar{p}_0 + \lambda_1 \vec{d}$ $\bar{q}_2 = \bar{p}_0 + \lambda_2 \vec{d}$

   return "2 intersections @ $\bar{q}_1$ and $\bar{q}_2$".

end

3. (a) Do <u>not</u> commute: <u>counterexample</u> (algebraic)

Consider $S = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and $T = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$$ST = \begin{bmatrix} 2 & 0 & 2 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad ST\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}$$

$$TS = \begin{bmatrix} 2 & 0 & 1 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad TS\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

not equal!

(b) Do <u>not</u> commute: the previous counterexample suffices to prove this case, since uniform scaling is a special case of non-uniform scaling.

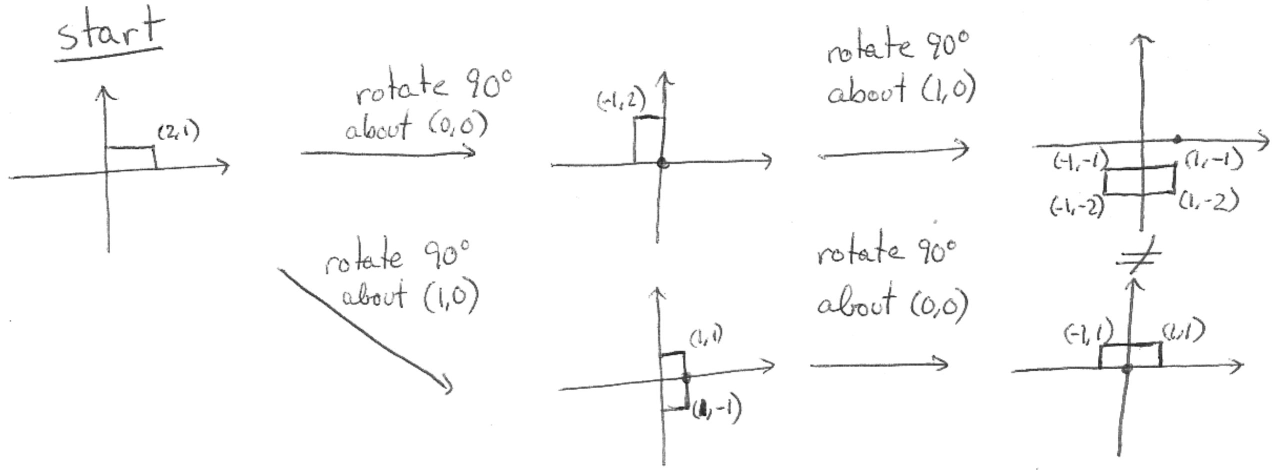
(c) Commute if the scaling is uniform, otherwise they do not.

~~Proof~~ Consider an arbitrary scaling transform $S = \begin{bmatrix} a & 0 & 1 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and an arbitrary rotation $R = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$$\Rightarrow SR = \begin{bmatrix} a\cos\theta & -a\sin\theta & 0 \\ b\sin\theta & b\cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad RS = \begin{bmatrix} a\cos\theta & -b\sin\theta & 0 \\ a\sin\theta & b\cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

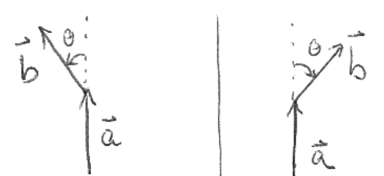
Notice that elements (1,1) and (2,2) of these matrices are equal, as are the translation and homogeneous parts of the matrix. Thus, these matrices will be equal if components (1,2) and (2,1) are equal. This is true when $b\sin\theta = a\sin\theta$ and $-a\sin\theta = -b\sin\theta$. Both conditions are satisfied only when $a=b$ (we consider the case $\sin\theta = 0$ trivial since it implies the rotation matrix is the identity). Therefore, the transforms commute if the scaling transform is uniform, otherwise they do not.

ⓓ Do NOT commute: geometric counterexample:



The final results are not equal, so the transforms do not commut

4.| Given two edges of the polygon, we can figure out which way the polygon edge is turning by using the right-hand rule of the cross product. Since the cross product only exists in 3 dimensions, we must extend the edge vectors ~~whi~~ with a zero in the third coordinate. Now, since both edge vectors lie in the xy-plane, the cross product ~~vi~~ vector will be perpendicula to this plane - thus only its z-component is non-zero.



$a_x b_y - a_y b_x > 0$
counter clockwise turn

$a_x b_y - a_y b_x < 0$
clockwise turn

~~Thus~~ Therefore, if the two edge vectors are $\vec{a}$ and $\vec{b}$, we only need to check the sign of the z-component of $\vec{a} \times \vec{b}$ to determine which way the edge is turning.

Note that if $(\vec{a} \times \vec{b}).z$ is zero, we get no information about convexity, since the vertices are colinear. However, it is not an invalid configuration.

This solution checks to see that every ~~pair o~~ triplet of consecutive edges is always turning in the same direction by checking that the product of the z-components of the cross products is always ≥ 0.

Algorithm/ Input: polygon vertices $\bar{v}_0, \bar{v}_1, \ldots, \bar{v}_n$

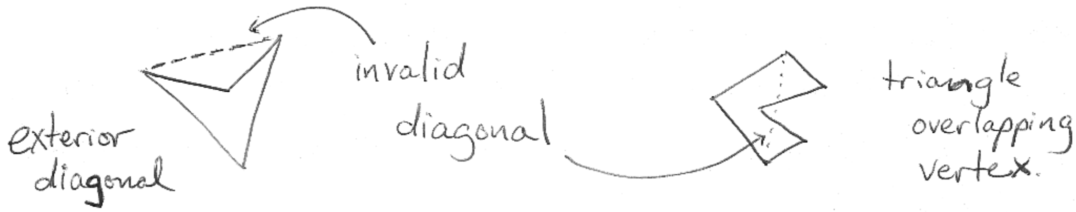
// In the following, all vertex indices should be taken
// mod (n+1).

$\vec{e}_1 \leftarrow \bar{v}_1 - \bar{v}_0$
$\vec{e}_2 \leftarrow \bar{v}_2 - \bar{v}_1$

$s_1 \leftarrow e_{1,x} e_{2,y} - e_{1,y} e_{2,x}$

convex ← true
$i \leftarrow 2$, $\vec{e}_1 \leftarrow \vec{e}_2$
while ($i \leq n$)
  $\vec{e}_2 \leftarrow \bar{v}_{i+1} - \bar{v}_i$
  $s_2 \leftarrow e_{1,x} e_{2,y} - e_{1,y} e_{2,x}$
  if ($s_1 s_2 < 0$)
    convex ← false
    break
  end
  $i \leftarrow i+1$
  $s_1 \leftarrow s_2$
  $\vec{e}_1 \leftarrow \vec{e}_2$
end
return convex.

Triangulating a convex polygon is simple, since we are guaranteed that every diagonal of the polygon is completely inside the polygon and does not intersect any polygon edges in the interior of the polygon. The latter constraint can be more clearly expressed by not allowing any polygon vertices to lie within the newly formed triangle.



The question then is whether we can always find a triangle in a concave polygon that satisfies the above two conditions. Luckily, the answer is yes — any polygon with four or more vertices has at least two such triangles, a result known as the "Two-Ear" theorem.

How can we test for each condition? Well, first we must know where the interior of the polygon is. In the following, I will assume the polygon vertices are given in counterclockwise order, so that the interior of the polygon is to the "left" when standing at vertex $\bar{v}_i$ looking toward $\bar{v}_{i+1}$. This means that the interior diagonals between $\bar{v}_{i-1}$ & $\bar{v}_{i+1}$ will be the ones for which $(\bar{v}_i - \bar{v}_{i-1}) \times (\bar{v}_{i+1} - \bar{v}_i)$ has z-component > 0. The overlapping condition is tested by the answer to question 5, so we will assume we have that function available.

Thus our algorithm is as follows: if the polygon is convex, draw diagonals from one vertex to all the others. If ~~not~~ not, find a valid triangle, add it to the list, and triangulate the resulting polygon.

Algorithm / Input: List of counter-clockwise vertices $\{\bar{v}_0, \bar{v}_1, \ldots, \bar{v}_n\}$
List of triangles T (in/out param)

if (isConvex($\{\bar{v}_0, \bar{v}_1, \ldots, \bar{v}_n\}$))
    ~~for~~ $i \leftarrow 1$
    while ($i < n$)
        T.add ($\{\bar{v}_0, \bar{v}_i, \bar{v}_{i+1}\}$)
    end

↳ continue on next page

```
else
   //search for a valid triangle.
   i ← 0
   while (i ≤ n)   //remember, vertex indices are taken mod (n+1).
      if ([(v̄_i - v̄_{i-1}) x (v̄_{i+1} - v̄_i)].z > 0)  // diagonal inside polygon
         validTri ← true
         for j ∈ ({1,2,...,n} - {i-1, i, i+1})  //test vertices that are
                                                 // not part of the candidate ∠
            if (inTriangle(v̄_{i-1}, v̄_i, v̄_{i+1}, v̄_j) ≠ "OUTSIDE")
               validTri ← false
               break
            end
         end //for
         if (validTri)
            //triangle is valid, add to list & recurse.
            T.add({v̄_{i-1}, v̄_i, v̄_{i+1}})
            triangulate({v_0, v_1, ..., v_{i-1}, v_{i+1}, ..., v_n}, T)
            break //out of while loop
         end
      end
      i ← i+1
   end //while
end // if (isConvex)
```

---

5. For this problem, we can again use the sign of the z-component of a cross product to determine on which side of the triangle edge the test point p lies.

$\bar{p}$ is always to the left

$\bar{p}$ is always to the right

$\bar{p}$ is once to the left, twice to the right.

As the above examples show, $\bar{p}$ will be inside the triangle if it is always to the right or always to the left of all edges taken in order. If the z-component of the cross product is ever 0, we know then that $\bar{p}$ and the vertices defining the edge are ~~all~~ collinear. In this case, we must determine whether $\bar{p}$ is between the vertices. We can do this by solving $\bar{p} = \bar{v}_i + \lambda(\bar{v}_{i+1} - \bar{v}_i)$ for λ and making sure $0 \leq \lambda \leq 1$.

Algorithm / Input: vertices $\bar{v}_0, \bar{v}_1, \bar{v}_2$, test point $\bar{p}$.

$c0 \leftarrow [(\bar{v}_1 - \bar{v}_0) \times (\bar{p} - v_0)].z$
$c1 \leftarrow [(\bar{v}_2 - \bar{v}_1) \times (\bar{p} - v_1)].z$
$c2 \leftarrow [(\bar{v}_0 - \bar{v}_2) \times (\bar{p} - \bar{v}_2)].z$

if $(c0 = 0)$ return checkEdge$(\bar{v}_0, \bar{v}_1, \bar{p})$ //see below
if $(c1 = 0)$ return checkEdge$(\bar{v}_1, \bar{v}_2, \bar{p})$
if $(c2 = 0)$ return checkEdge$(\bar{v}_2, \bar{v}_0, \bar{p})$
if $[(c0 < 0$ and $c1 < 0$ and $c2 < 0)$
<u>or</u> $(c0 > 0$ and $c1 > 0$ and $c2 > 0)]$
return INSIDE
else
return OUTSIDE
end

---

checkEdge$(\bar{v}_a, \bar{v}_b, \bar{p})$
if $(V_{b,x} - V_{a,x} = 0)$ //this works since we are
$\lambda = \dfrac{p_y - V_{a,y}}{V_{b,y} - V_{a,y}}$ //assuming non-degenerate triangles
else //so $\bar{V}_a \neq \bar{V}_b$
$\lambda = \dfrac{p_x - V_{a,x}}{V_{b,x} - V_{a,x}}$
end
if $(\lambda \geq 0$ and $\lambda \leq 1)$
return ON_EDGE
else
return OUTSIDE
end

Marking scheme:

1. 2 points for each answer.
2. I wanted to see full work. “Solving for lambda” was not a valid answer, but “this is a quadratic function of lambda with coefficients.... which we can solve using the quadratic equation” was. In particular, I was looking for the quadratic coefficients and for a concrete statement of how to determine the number of solutions, the locations of the intersection points, and a brief algorithm.
3. 2 points each for a,b,d, 4 points for c. For full marks in c, I wanted a proof that the transforms commute if the scaling is uniform.
4. 7 points per algorithm. I was picky here. For example, if you wanted to check the interior angle, you had to show me how you know which angle between two lines is interior. Non-trivial computations that were not developed fully lost marks. I also took marks off if I was able to find a polygon which your algorithm would fail on. Very inefficient algorithms lost 1 point.
5. 5 points for inside/outside test, 1 point if your algorithm returns “on_edge” in certain cases, 4 points for the correct “on_edge” test.