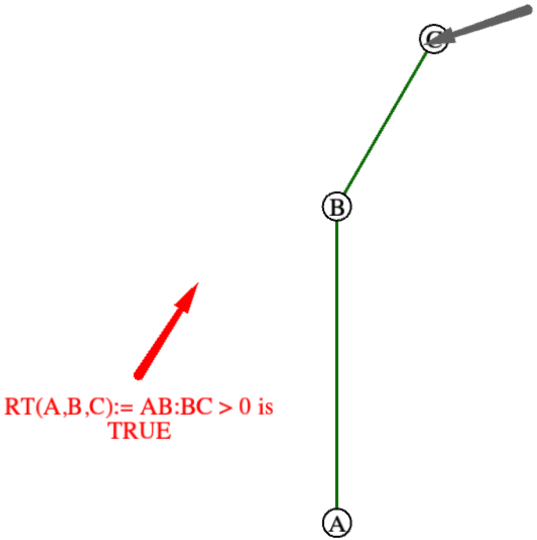
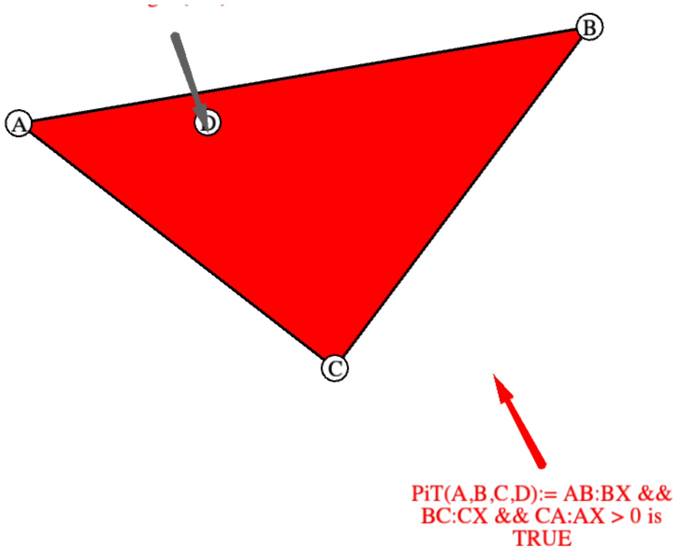
4 EXERCISES ON POINTS AND VECTORS





CS3451 FALL 2020 PROJECT 2

Saumya JAIN

Phase 1: PROBLEM = Right turn test

Given three points A, B and C, and their resulting line segments AB and BC, correctly return if the segments make a right turn at the point B.

COMMENTS:

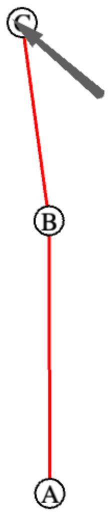
In order to accurately state if the line segments make a right turn, we assume the following:

i. The set of points {A, B, C} are labeled contiguously.
ii. A right turn is essentially any case in which the segment BC deviates to the right of the direction/heading of AB.

Here, there is no ambiguity.

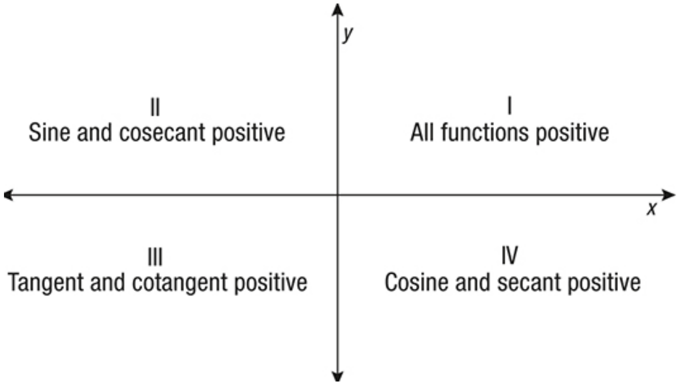
Phase 1: Solution math

We can compute if the segment BC makes a right turn or not through simple geometry and trigonometry.



- First, we make vectors out of line segments AB and BC for ease of calculations. So we have, vector **AB** and vector **BC**.

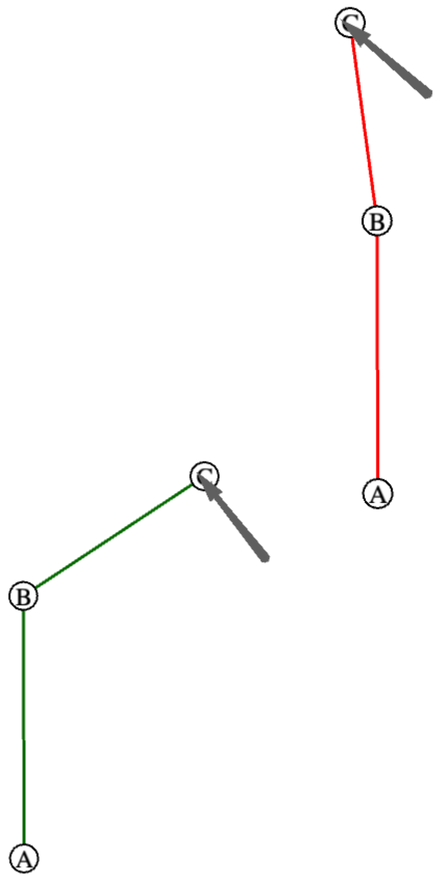
- We know that the value of the sine function is positive for angles values of 0 - 180 degrees (0 - π radians) . We also know that the det product of **AB** and **BC** is calculated by scalar |**AB**| |**BC**| sin(**AB** ^ **BC**).



Phase 1: Solution math

- The calculation of det product factors in the value of the sine function of the angle between vectors **AB** and **BC**. So, the det product will have the same sign (+ve or –ve) as the result of the sine function.
- The det product of the vectors will be:
    - Positive: For angles 0 – 180 degrees (0 - π radians)
    - Negative: For angles 180-360 degrees (π - 2π radians)

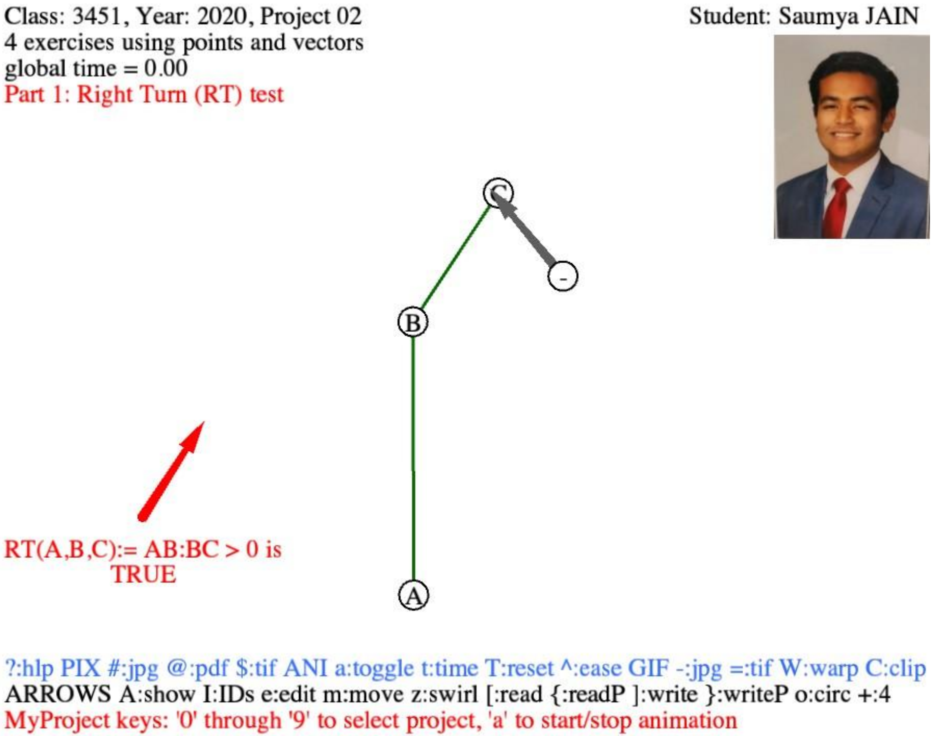
- Hence, if BC turns right, we know that its det product is positive. We can use this to correctly determine the result every time. We calculate det product of **AB** and **BC** and check if it is greater than or less than 0.



Phase 1: Code and whole screen shot of result

```
//========================================================================= PART 1
void showPart1(PNT A, PNT B, PNT C, PNT D) //
  {
  PartTitle[1] =   "Right Turn (RT) test"; // https://en.wikipedia.org/wiki/Parallelogram
  // To position the text of your solution on the canvas:
  // Place your mouse-arrow and press 'C', which prints 2 lines in the botom pane. Copy then here:
  StartClip = P(143,493);
  EndClip = P(197,409);
  // Add  your solution  to the MyText String below
  MyText="RT(A,B,C):= AB:BC > 0";
  PNT X = Mouse();
  if(animate) X = P(C);
  if(RT(A,B,X)) { MyText = MyText + " is TRUE"; cwF(dgreen,3); }
  else { MyText = MyText + " is FALSE"; cwF(dred,3); }
  show(A,B,X);
  guide="MyProject keys: '0' through '9' to select project, 'a' to start/stop animation ";
  if(showIDs) { A.circledLabel("A"); B.circledLabel("B"); X.circledLabel("C"); } // D.circledLabel("D");
  }

boolean RT(PNT A, PNT B, PNT C) {
  // Making vectors out of both line segments
  VCT AB = V(A, B);
  VCT BC = V(B, C);
  // Calculating det product of vectors to check sign of magnitude
  float detOfVectors = det(AB, BC);
  return detOfVectors > 0;
} // EDIT THIS
```

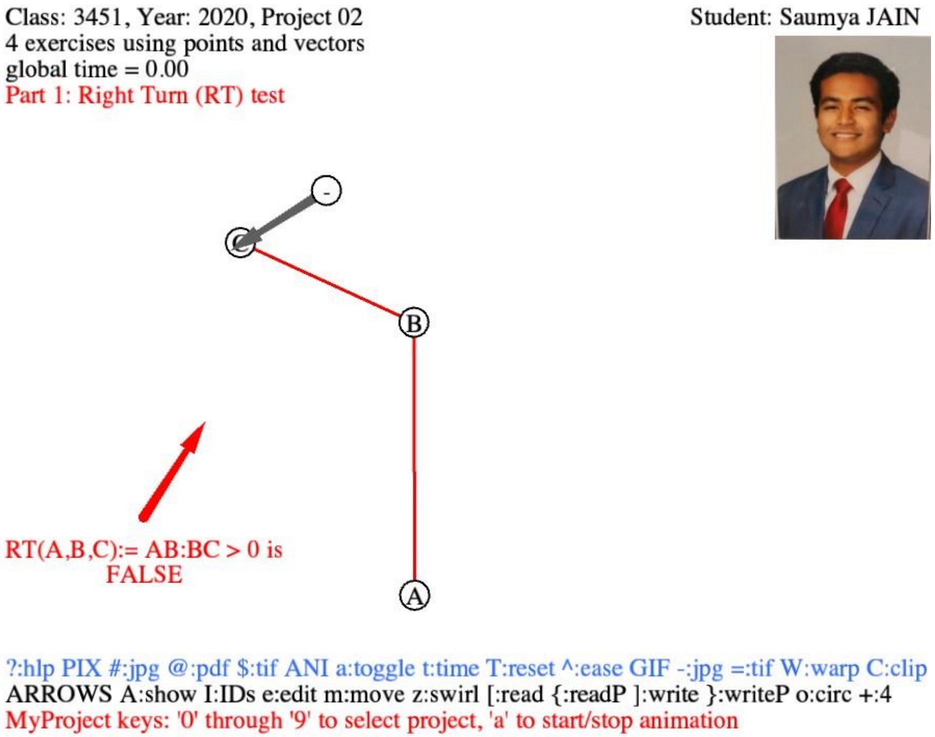
Phase 1: Results and limitations of your implementation (Right Turn Passed)



DISCLAIMER, LIMITATIONS:

This solution always works, regardless of the orientations of **AB** and **BC**

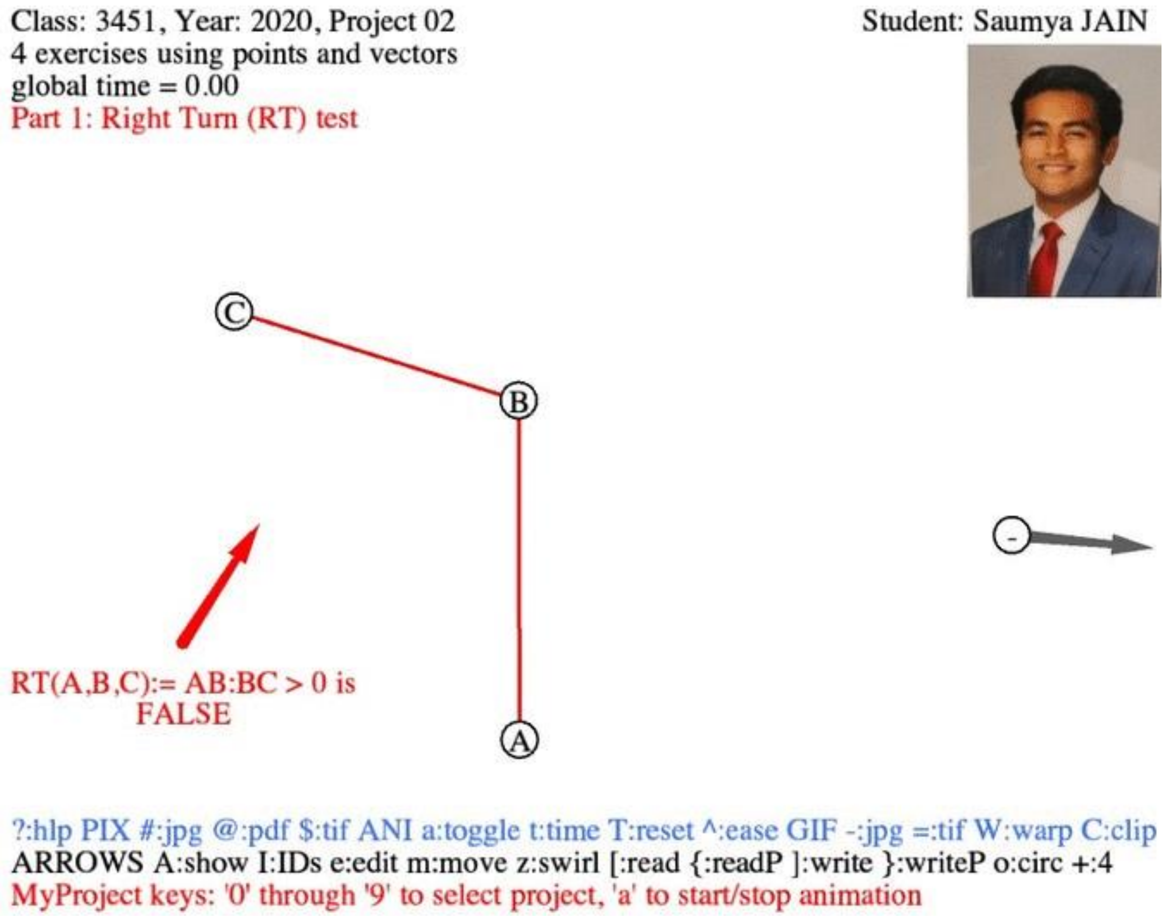
Phase 1: Results and limitations of your implementation (Right Turn Not Passed)



DISCLAIMER, LIMITATIONS:

This solution always works, regardless of the orientations of **AB** and **BC**

Phase 1: GIF



Class: 3451, Year: 2020, Project 02
4 exercises using points and vectors
global time = 0.00
Part 1: Right Turn (RT) test
Student: Saumya JAIN
C
B
A
-
RT(A,B,C):= AB:BC > 0 is
FALSE
?:hlp PIX #:jpg @:pdf $:tif ANI a:toggle t:time T:reset ^:ease GIF -:jpg =:tif W:warp C:clip
ARROWS A:show I:IDs e:edit m:move z:swirl [:read {:readP]:write }:writeP o:circ +:4
MyProject keys: '0' through '9' to select project, 'a' to start/stop animation

Phase 2: PROBLEM = Point-in-Triangle test

Given three points A, B and C which compose a triangle, and a fourth point D, correctly return if D is within the triangle.

COMMENTS:

In order to accurately state if the fourth point is within the triangle, we assume the following:

i. The initial points {A, B, C} form a triangle.

ii. The set of points {A, B, C} are ordered clockwise or anticlockwise around the triangle.

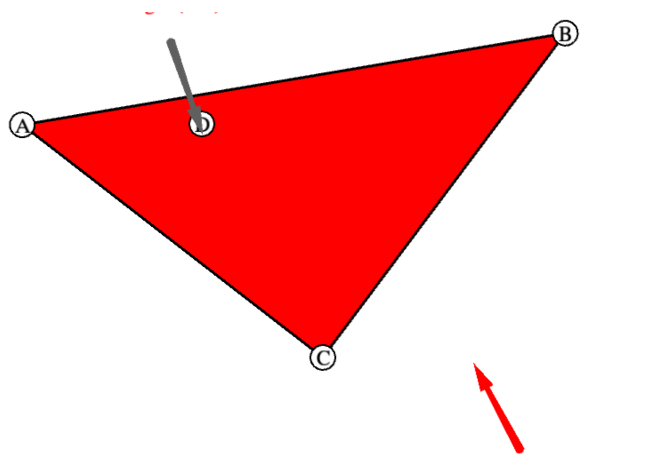
iii. Point D is said to be within the triangle if it lies within the boundary of triangle ABC. It should be contained by sides AB, BC and CA.

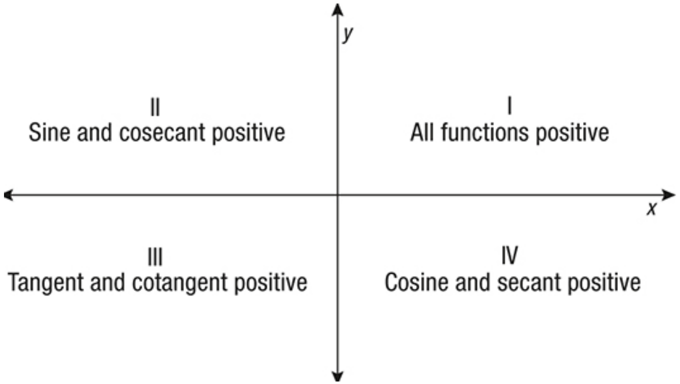
Phase 2: Solution math

We can compute if the point D is within triangle ABC through simple geometry and trigonometry. We can use our solution from Phase 1 (Right Turn Test) to efficiently calculate this solution.

- First, we make vectors out of triangle edges AB, BC and CA for ease of calculations. So we have, vector **AB,** vector **BC** and vector **CA**.

- Using the mathematics from the right line test, we know that We know that the value of the sine function is positive for angles values of 0 - 180 degrees (0 - π radians) . So, the det product will have the same sign (+ve or –ve) as the result of the sine function.



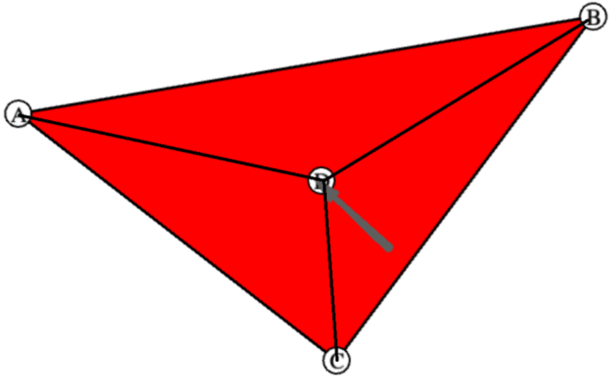


Phase 2: Solution math

- Here, we can see that if point D lies within the triangle, we can make resulting line segments AD, BD and CD.

- The pairs of line segments (AB and BD), (BC and CD) and (CA and AD) all pass the right line test when D lies inside the triangle. We can use this to our advantage in code.

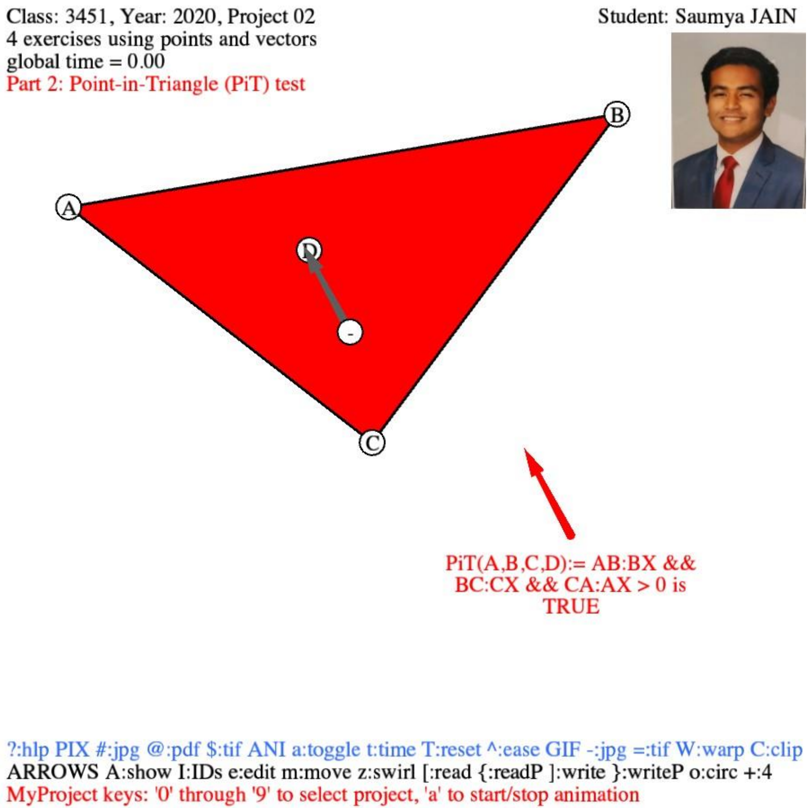
- Hence, since these segments turn right, we know that its det product is positive. We calculate det product of for all three pairs of segments and check if it is greater than or less than 0. If it is positive, we know that point D lies inside the triangle.



Phase 2: Code and whole screen shot of result

```
//==================================================================== PART 2
void showPart2(PNT A, PNT B, PNT C, PNT D) //
  {
  PartTitle[2] = "Point-in-Triangle (PiT) test";
  StartClip = P(638,589);
  EndClip = P(591,501);
  // Add  your solution  to the MyText String below
  MyText="PiT(A,B,C,D):= AB:BX && BC:CX && CA:AX > 0";
  PNT X = Mouse();
  if(animate) X = P(D);
  if(PiT(X,A,B,C)) { MyText = MyText + " is TRUE"; cwf(black,3,red); }
  else { MyText = MyText + " is FALSE"; cwf(black,3,green); }
  showLoop(A,B,C);
  guide="MyProject keys: '0' through '9' to select project, 'a' to start/stop animation ";
  if(showIDs) { A.circledLabel("A"); B.circledLabel("B"); C.circledLabel("C");  X.circledLabel("D"); }
  }

boolean PiT(PNT X, PNT A, PNT B, PNT C) {
  // Calculating if all sides of triangle make right turns with the point (if True, it is within triangle)
  return (RT(A, B, X) && RT(B, C, X) && RT(C, A, X));
  } // EDIT THIS

```

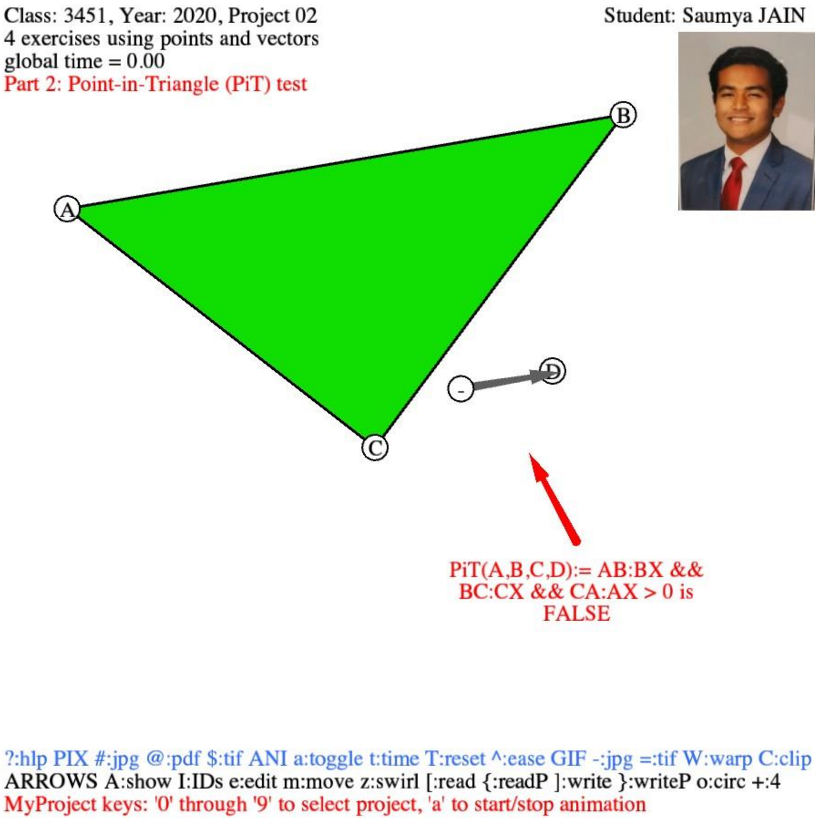
Phase 2: Results and limitations of your implementation (Point in Triangle)



DISCLAIMER, LIMITATIONS:

This solution always works, even in edge cases where point D is on an edge or vertex of triangle ABC.

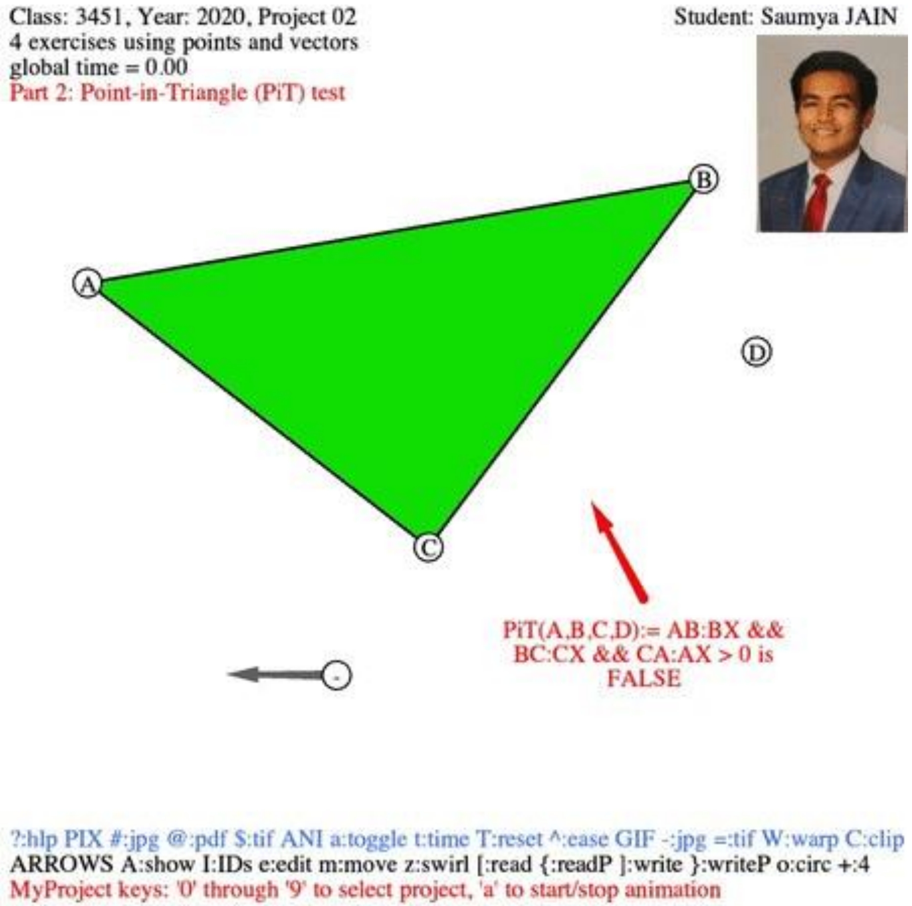
Phase 2: Results and limitations of your implementation (Point in Triangle)



DISCLAIMER, LIMITATIONS:

This solution always works, even in edge cases where point D is on an edge or vertex of triangle ABC.

Phase 2: GIF


Class: 3451, Year: 2020, Project 02
4 exercises using points and vectors
global time = 0.00
Part 2: Point-in-Triangle (PiT) test
Student: Saumya JAIN
A
B
C
D
PiT(A,B,C,D):= AB:BX &&
BC:CX && CA:AX > 0 is
FALSE
?:hlp PIX #:jpg @:pdf $:tif ANI a:toggle t:time T:reset ^:ease GIF -:jpg =:tif W:warp C:clip
ARROWS A:show I:IDs e:edit m:move z:swirl [:read {:readP]:write }:writeP o:circ +:4
MyProject keys: '0' through '9' to select project, 'a' to start/stop animation

Phase 3: PROBLEM = Edge/Edge intersection

Given 2 edges (AB and CD) determine the following:

i. whether AB and CD intersect

ii. where the point of intersection X lies

COMMENTS:

In order to accurately state if AB and CD intersect and the location of their point of intersection, we assume the following:

i. Edge AB is composed of points A and B.

ii. Edge CD is composed of points C and D.

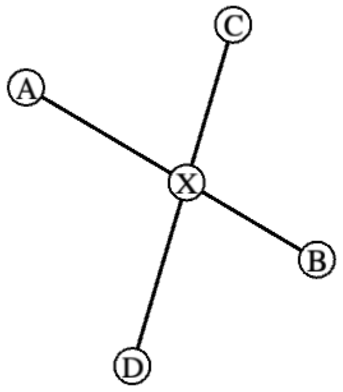
iii. The point of intersection X is visible only when AB and CD intersect.

iv. The edges AB and CD are never parallel.

Here, there can be the ambiguity of the orientation of the edges with respect to each other. Edges can exist as AB/BA and CD/DC. For simplicity, I have chosen the orientation AB and CD. The order of points of the edges has no effect on our result.

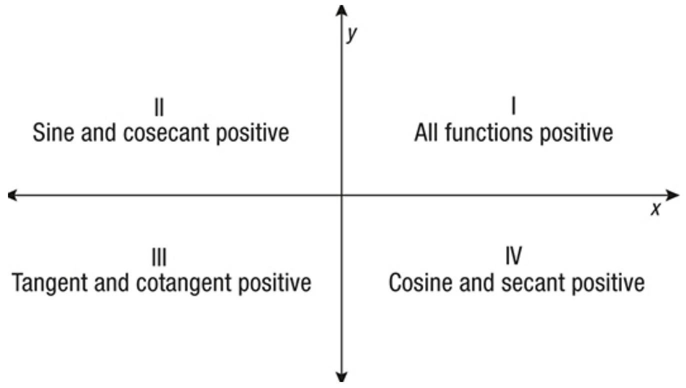
Phase 3: Solution math (Part 1: Do edges intersect?)

We can compute if the edges AB and CD intersect through simple geometry and trigonometry. We can use the properties of dot products to prove our solution.



- Let us assume there is a point of intersection X of edges AB and CD. We make vectors out of the resulting segments XA, XB, XC and XD. So we have vector **XA,** vector **XB,** vector **XC** and vector **XD**.

- We know that the value of the cosine function is negative for angle values of 90 - 270 degrees (π/2 - 3π/2 radians) . We also know that the dot product of anu vectors **AB** and **BC** is calculated by scalar |**AB**| |**BC**| cosine(**AB** ^ **BC**).

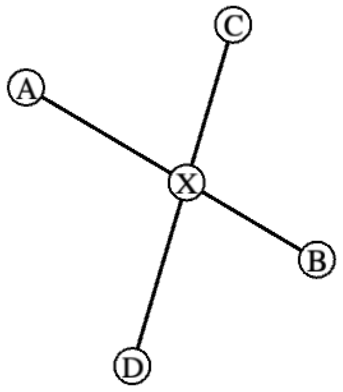


Phase 3: Solution math (Part 1: Do edges intersect?)

- The calculation of dot product factors in the value of the cosine function of the angle between any 2 vectors. So, the dot product will have the same sign (+ve or –ve) as the result of the cosine function.

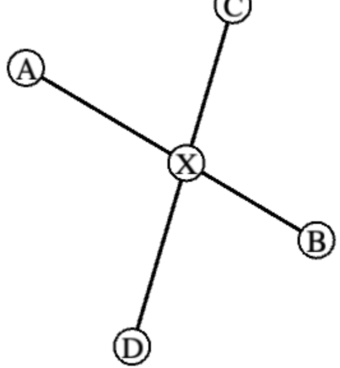
- The dot product of the vectors will be:
  - Positive: For angles 0 – 90 and 270 - 360 degrees (0 - π and 3π/2 - 2π radians)
  - Negative: For angles 90 - 270 degrees (π/2 - 3π/2 radians)

- Now, we can calculate dot products of vectors **XA** and **XB.** If X lies on the AB line, the dot product will be negative since angle will be 180 degrees (π radians). Similarly, if X lies on CD dot products of **XC** and **XD** will also be negative since angle will be 180 degrees (π radians).



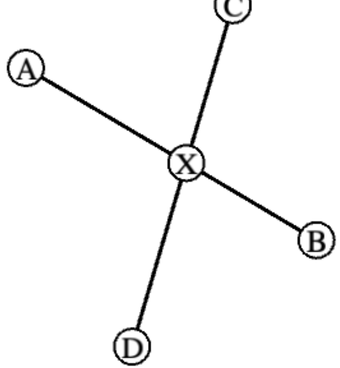
Phase 3: Solution math (Part 1: Do edges intersect?)

- If the dot products of **(XA and XB)** and **(XC and XD)** simultaneously evaluate to a negative value, we know that X lies on both AB and CD. This is only true when AB and CD intersect.



Phase 3: Solution math (Part 2: Point of Intersection?)

We can compute the point of intersection of edges AB and CD easily.



- We make vectors out of edges AB and CD. So we have vector **AB** and vector **CD**.

- Let the line equation of **AB** be L1 = A + xAB and line equation of **CD** be L2 = C + yCD.

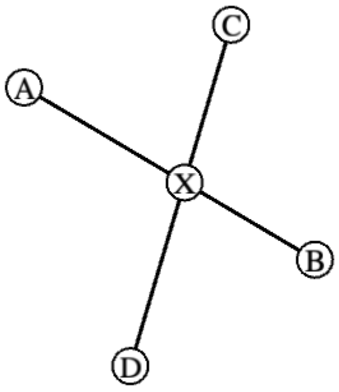
- To find the point of intersection, we can write:

  A + xAB = C + yCD

Phase 3: Solution math (Part 2: Point of Intersection?)

- Further rearranging this equation:

  A + xAB = C + yCD
  xAB = (C – A) + yCD (C-A represents a vector **AC**)
  xAB = **AC** + yCD



- Applying det product to both sides, we get:

  xAB : CD = AC : CD
  x = AC : CD / AB : CD

- So, our point of intersection is:
  X = A + (AC : CD / AB : CD) * AB

- (AC : CD / AB : CD) is used to scale up AB on A.

Phase 3: Code and whole screen shot of result

```
//=========================================================================== PART 3
void showPart3(PNT A, PNT B, PNT C, PNT D) //
  {
  PartTitle[3] =   "Edge-Edge Intersection test (ExE) and point"; // https://pin.it/7wgYDeq
  StartClip = P(156,632);
  EndClip = P(175,534);  // Add  your solution  to the MyText String below
  MyText="ExE(A,B,C,D):= XA•XB && XC•XD < 0 (X is point of intersection)";
  cwf(black,3,yellow); show(A,B); show(C,D);
  if(ExE(A,B,C,D))
    {
    PNT X = LiL(A,B,C,D);
    cwf(blue,1,blue); show(X,6);
    if(showIDs) X.circledLabel("X");
    MyText=MyText+" is TRUE";
    } else {
      MyText=MyText+" is FALSE";
    }
  guide="MyProject keys: '0' through '9' to select project, 'a' to start/stop animation ";
  if(showIDs) { A.circledLabel("A"); B.circledLabel("B"); C.circledLabel("C");  D.circledLabel("D"); }
  }
boolean  ExE(PNT A, PNT B, PNT C, PNT D)
  {
    // Calculate the would be intersection point of AB and CD
    PNT X = LiL(A, B, C, D);
    // If the point lies on both AB and CD, the segments intersect
    // The value of the dot products of the segments produced will be negative if an intersection happens
    boolean pointOnAB = dot(V(X, A), V(X, B)) < 0;
    boolean pointOnCD = dot(V(X, C), V(X, D)) < 0;
    return pointOnAB && pointOnCD;
  }

```

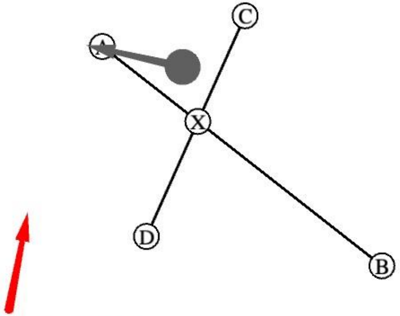
Phase 3: Code and whole screen shot of result

```
PNT LiL(PNT A, PNT B, PNT C, PNT D)
  {
    // Making vectors of segments AB and CD
    VCT AB = V(A, B);
    VCT CD = V(C, D);
    float detOfABCD = det(AB, CD);
    float detOfACCD = det(V(A, C), CD);
    float scale = detOfACCD/detOfABCD;
    // Calculating point of intersection by scaling up AB
    PNT X = P(A, AB.scaleBy(scale));
    return X; // EDIT THIS
  }

```

Phase 3: Results and limitations of your implementation (Intersection)

Class: 3451, Year: 2020, Project 02
4 exercises using points and vectors
global time = 0.00
Part 3: Edge-Edge Intersection test (ExE) and point

Student: Saumya JAIN



ExE(A,B,C,D):= XA•XB && XC•XD < 0 (X is point of intersection) is TRUE

?:hlp PIX #:jpg @:pdf $:tif ANI a:toggle t:time T:reset ^:ease GIF -:jpg =:tif W:warp C:clip
ARROWS A:show I:IDs e:edit m:move z:swirl [:read {:readP]:write }:writeP o:circ +:4
MyProject keys: '0' through '9' to select project, 'a' to start/stop animation

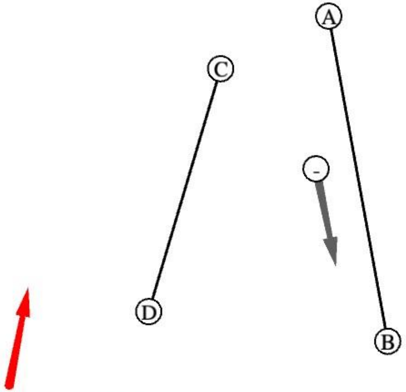
DISCLAIMER, LIMITATIONS:

This solution always works, regardless of the orientations of AB and CD.

Phase 3: Results and limitations of your implementation (No Intersection)

Class: 3451, Year: 2020, Project 02
4 exercises using points and vectors
global time = 0.00
Part 3: Edge-Edge Intersection test (ExE) and point

Student: Saumya JAIN



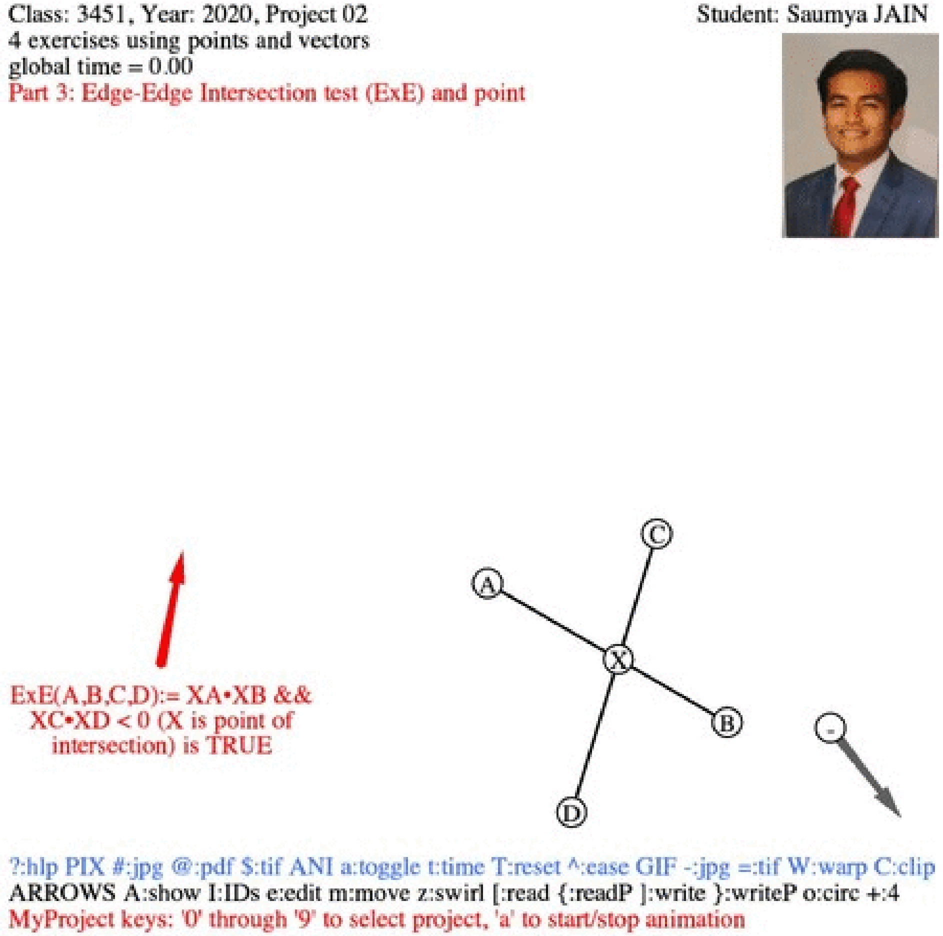
ExE(A,B,C,D):= XA•XB &&
XC•XD < 0 (X is point of
intersection) is FALSE

?:hlp PIX #:jpg @:pdf $:tif ANI a:toggle t:time T:reset ^:ease GIF -:jpg =:tif W:warp C:clip
ARROWS A:show I:IDs e:edit m:move z:swirl [:read {:readP]:write }:writeP o:circ +:4
MyProject keys: '0' through '9' to select project, 'a' to start/stop animation

DISCLAIMER, LIMITATIONS:

This solution always works, regardless of the orientations of AB and CD.

Phase 3: GIF


Class: 3451, Year: 2020, Project 02
4 exercises using points and vectors
global time = 0.00
Part 3: Edge-Edge Intersection test (ExE) and point
Student: Saumya JAIN
C
A
X
B
D
ExE(A,B,C,D):= XA•XB &&
XC•XD < 0 (X is point of
intersection) is TRUE
?:hlp PIX #:jpg @:pdf $:tif ANI a:toggle t:time T:reset ^:ease GIF -:jpg =:tif W:warp C:clip
ARROWS A:show I:IDs e:edit m:move z:swirl [:read {:readP]:write }:writeP o:circ +:4
MyProject keys: '0' through '9' to select project, 'a' to start/stop animation

Phase 4: Closest projection on triangle border

Given three points A, B and C which compose a triangle, and a fourth point D, correctly return the point on the perimeter of the triangle closest to D.

COMMENTS:

In order to accurately return the point of closest projection, we assume the following:

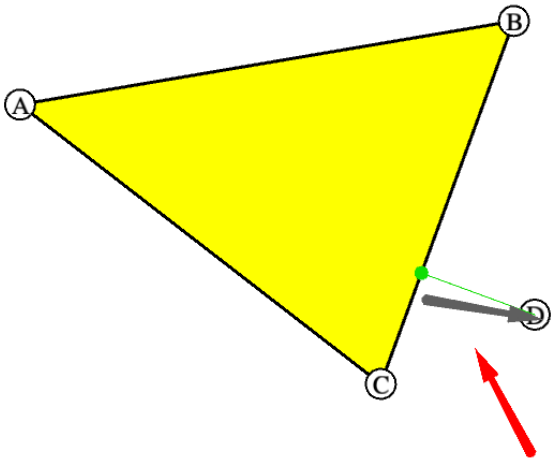
i. The initial points {A, B, C} form a triangle.

ii. The set of points {A, B, C} are ordered clockwise or anticlockwise around the triangle.

iii. Point D can have its point of closest projection on an edge or vertex of the triangle.

Here, there is no ambiguity.

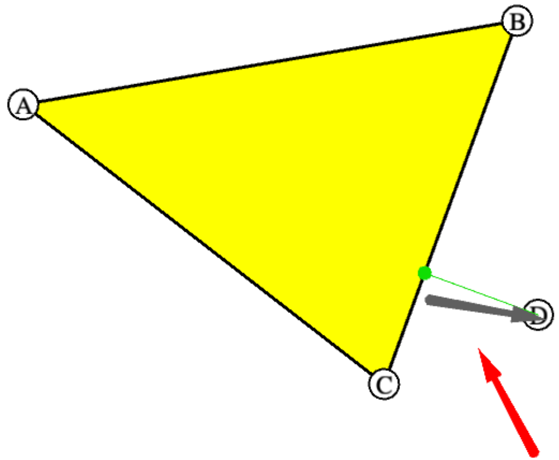
Phase 4: Solution math



We can compute the point on the triangle closest to D through simple vector operations and geometry. We can also use our previous functions to help us.

- We can start by computing the distance to each edge (AB, BC, CA) and each vertex (A, B, C) of the triangle from point D. Our desired point will be whichever distance is the shortest.
- To calculate the point of projection from D to edge AB:
  - First, we make a vector **AB** from edge AB. We also make a vector between points A and D as **AD**.
  - Next, we can find the projection vector of **AB** and **AD** by scaling **AB** by the dot product of **AB** and **AD** normalized by magnitude of **AB**. This is the correct projection vector as we are finding how much **AD** goes in the direction of **AB**.

Phase 4: Solution math



- The point formed by this projection lies along AB and gives us the shortest perpendicular distance from D to **AB**. From math, we know that the shortest distance from a point to a line is the perpendicular distance in between.
- Using the inbuilt distance functions, we can calculate the distance between D and every such projection calculated for each side and vertex of the triangle and return the one which produces the shortest distance.
- I utilized the ExE function from Phase 3 to find the point of intersections of each edge and the edge formed by joining D and its projection. This gives us the points needed to calculate each case’s distance and find the optimal one.

Phase 4: Code and whole screen shot of result

```
void showPart4(PNT A, PNT B, PNT C, PNT D) //
  {
  PartTitle[4] = "Projection on Triangle (POT)";
  StartClip = P(638,589);
  EndClip = P(591,501);
  // Add  your solution  to the MyText String below
  MyText="PiT(A,B,C,D):= see slides";
  PNT X = Mouse();
  if(animate) X = P(D);
  cwf(black,3,yellow); showLoop(A,B,C);
  cwf(blue,1,blue); show(X,6);
  PNT P = PoT(X,A,B,C);
  guide="MyProject keys: '0' through '9' to select project, 'a' to start/stop animation ";
  if(showIDs) { A.circledLabel("A"); B.circledLabel("B"); C.circledLabel("C");  X.circledLabel("D"); }
  if(PiT(X,A,B,C)) cwf(red,1,red); else cwf(green,1,green);
  show(P,6); show(P,X);
  }
PNT PoT(PNT X, PNT A, PNT B, PNT C)
  {
    // Find each projection
    PNT projectionAB = projection(X, A, B);
    PNT projectionBC = projection(X, B, C);
    PNT projectionCA = projection(X, C, A);
    PNT shortestDistance = A;

```

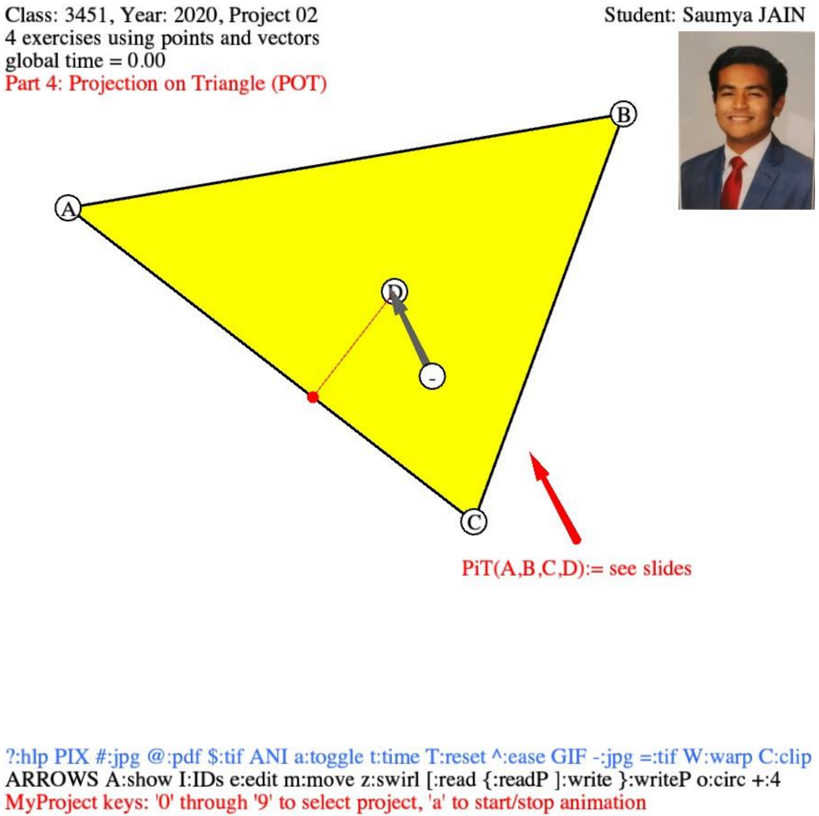
Phase 4: Code and whole screen shot of result

```
    // If cases to check which case produces the least distance
    if(d(X, B) < d(X, shortestDistance)) {
      shortestDistance = B;
    }
    if(d(X, C) < d(X, shortestDistance)) {
      shortestDistance = C;
    }

    if (ExE(A, B, X, P(projectionAB, V(X, projectionAB))) && d(X, projectionAB) < d(X, shortestDistance)) {
      shortestDistance = projectionAB;
    }
    if (ExE(B, C, X, P(projectionBC, V(X, projectionBC))) && d(X, projectionBC) < d(X, shortestDistance)) {
      shortestDistance = projectionBC;
    }
    if (ExE(C, A, X, P(projectionCA, V(X, projectionCA))) && d(X, projectionCA) < d(X, shortestDistance)) {
      shortestDistance = projectionCA;
    }

    return shortestDistance;
    // edit this, feel free to use other helper functions to make your code simple and elegant (not necessarily fastest)
  }
```

Phase 4: Code and whole screen shot of result

```
// Helper function that calculates the point projected from D to the edge in the parameter
PNT projection(PNT X, PNT A, PNT B) {
    VCT AB = V(A, B);
    float dotOfABAX = dot(AB, V(A, X));
    VCT projectionVector = AB.scaleBy(dotOfABAX / (n(AB) * n(AB)));
    PNT projectionPoint = P(A, projectionVector);
    return projectionPoint;
  }

```

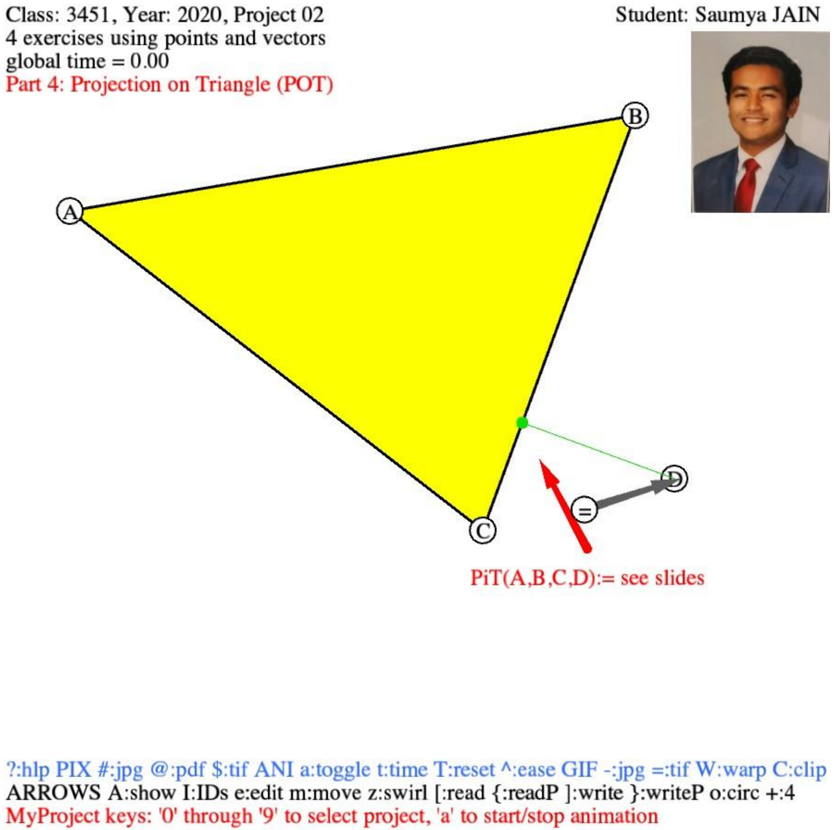
Phase 4: Results and limitations of your implementation (Point is Inside)



DISCLAIMER, LIMITATIONS:

This solution always works, regardless of the orientations of AB and CD.

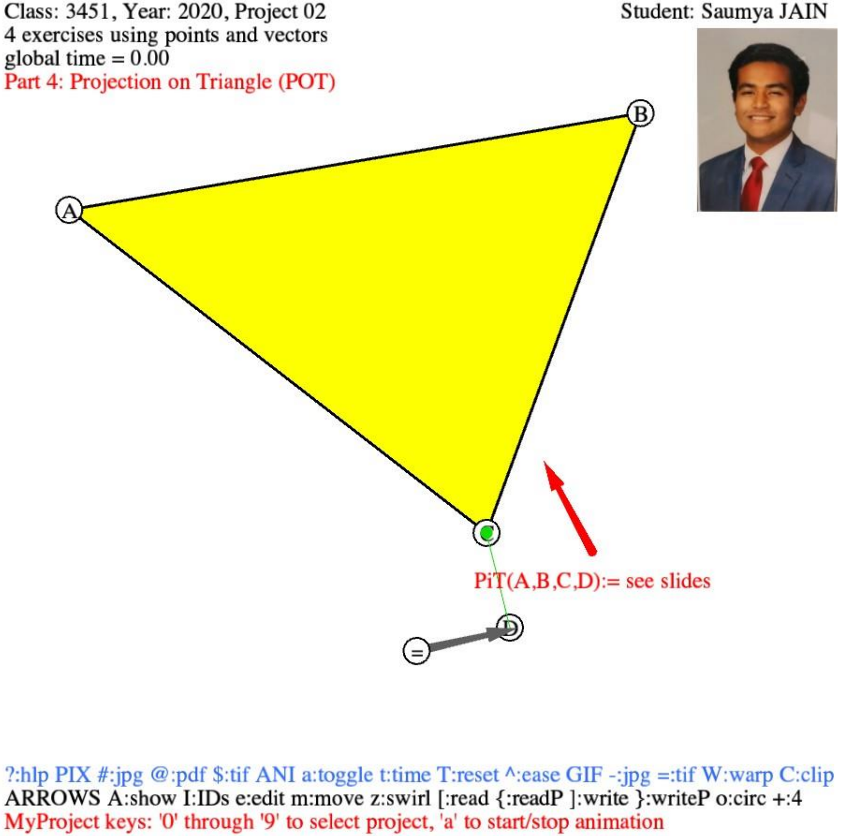
Phase 4: Results and limitations of your implementation (Point is Outside - Edge)



DISCLAIMER, LIMITATIONS:

This solution always works, regardless of the orientations of AB and CD.

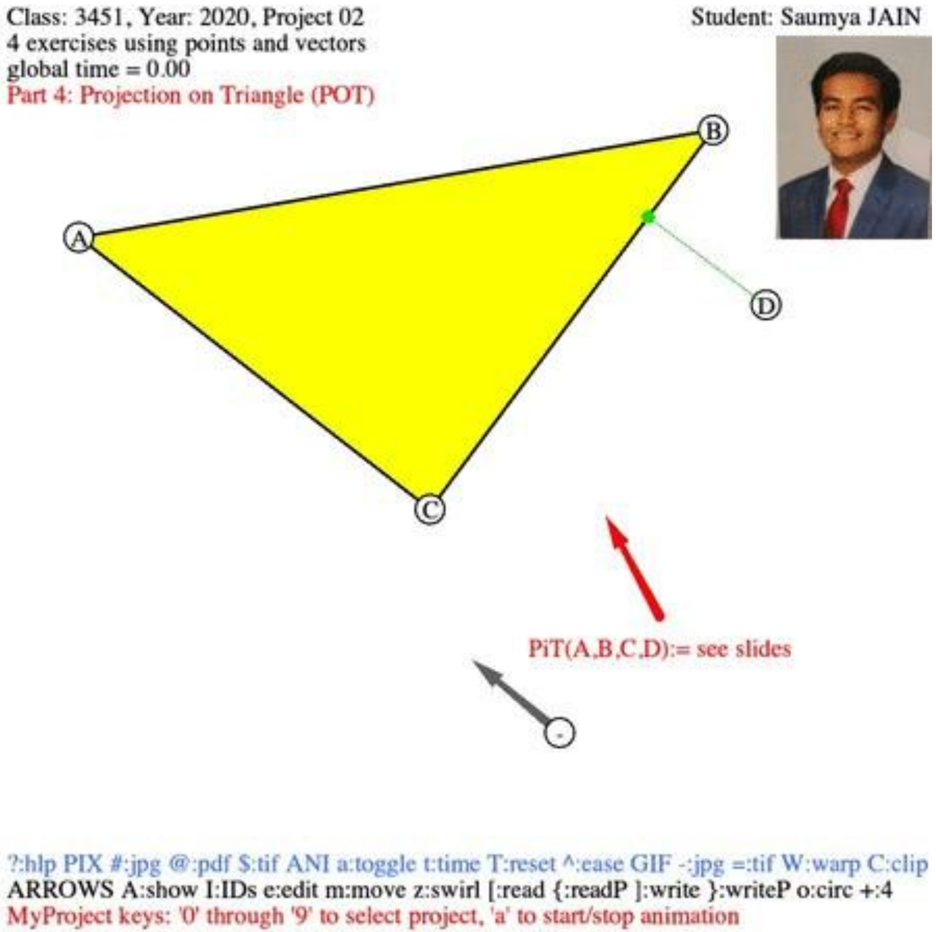
Phase 4: Results and limitations of your implementation (Point is Outside - Vertex)



DISCLAIMER, LIMITATIONS:

This solution always works, regardless of the orientations of AB and CD.

Phase 4: GIF


Class: 3451, Year: 2020, Project 02
4 exercises using points and vectors
global time = 0.00
Part 4: Projection on Triangle (POT)
Student: Saumya JAIN
A
B
C
D
PiT(A,B,C,D):= see slides
?:hlp PIX #:jpg @:pdf $:tif ANI a:toggle t:time T:reset ^:ease GIF -:jpg =:tif W:warp C:clip
ARROWS A:show I:IDs e:edit m:move z:swirl [:read {:readP]:write }:writeP o:circ +:4
MyProject keys: '0' through '9' to select project, 'a' to start/stop animation