Table of Contents

Intersection of a line segment with a triangle

Determination of parts within, on and outside the triangle

Rohit Sud (rohit@gatech.edu) and Abhishek Venkatesh (venky@gatech.edu)

Problem statement

Given a triangle ABC and a line segment PQ, determine the region where the line segment intersects the triangle. Further classify the portions of the line segment as lying outside the triangle, on the triangle and inside the triangle and colour them with a green, orange and red stroke respectively.

Introduction

In this report, we describe a method to perform clipping of a line segment with respect to a triangle in 2D vector space. The number of end-points of the line segment inside the triangle is determined first. Based on the results we calculate the intersection(s) of the line segment with the edges of the triangle. Theoretically the intersection is a single point but because of the pixel based display we see that there is an overlapping segment around the point of intersection. We find an approximate extent of the overlap (orange colour) using the Algorithm 4 described later.

Additionally we also show how to use the above method to calculate the intersection between two triangles, the union, intersection and difference of two triangles.

Colour coding of the resultant line segment is as follows: The part of line segment shown in green is outside the triangle, the part of line segment shown in red is inside the triangle and the part of line segment overlapping with an edge of the clipping triangle is shown in orange.

Algorithm to find the intersection of a line segment with a triangle

Let PQ be the line segment and ABC be the triangle. Both are assumed to be in the same 2D plane. We first identify how many points of the line segment PQ lie inside the triangle (edges and vertices of the triangle are treated as outside). This is done by using Algorithm 1 on both the points of PQ. There are 3 cases and the number of intersections varies with each case as shown below:

Case 1(Both points of the line segment PQ are inside the triangle): This is a trivial case. Since both points of the line segment are inside the triangle there can be no intersection. So the line segment PQ is colour coded in red completely.

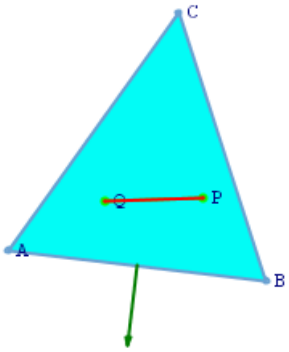

Figure 1

Case 2 (Both points are outside the triangle): This means that the segment must intersect the triangle at 2 points on different edges or at a common vertex or overlap with an edge (parallel with an edge and intersect it) or it doesn't intersect at all. Following steps are taken to process this case:

1. Determine if PQ intersects any edge of the triangle using Algorithm 2 repeatedly for the 3 edges of the triangle.
2. If there is no intersection, then this is a trivial case and the algorithm finishes and there is nothing left to be done. Sample output:

3. If PQ intersects with 2 edges of the triangle:

a) Find the points of intersection using algorithm 3. Let these be X_0 and X_1 (which may be equal if it is a vertex of the triangle).

b) Theoretically, line segments intersect at a unique point. But because of the pixel based display we see that the line segments actually have more than one point of intersection. To accommodate this and improve the visual feel we find the portion of line segment PQ around the intersection point which is very close to the corresponding edge of the triangle. This portion of line is determined using Algorithm 4 and coloured as orange to denote the intersection of PQ with the corresponding edge. This portion of line segment is denoted as a line segment defined by 2 points XS_1, XS_2 for X0 (and XS_3, XS_4 for X1). Colour the segments XS_1XS_2 and XS_3XS_4 as orange and the remainder line segment PQ using green colour.

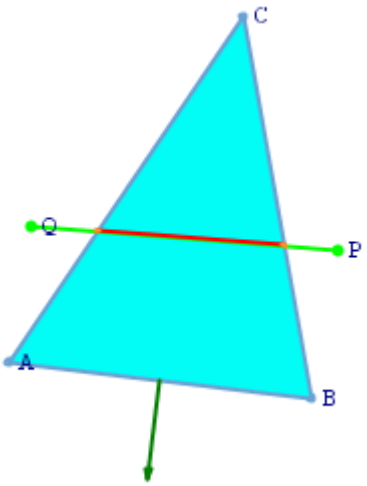

Figure 2

c) It might also happen that PQ intersects with an edge of the triangle and is also parallel to it. In that case we determine the common segment of PQ and the corresponding edge and colour that common segment as orange and the remainder of PQ in green. This will happen if Algorithm 2 tells us that the lines intersect but Algorithm 3 fails to find a point of intersection.

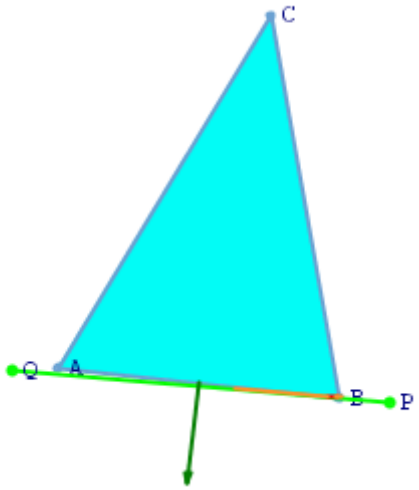

Figure 3

Case 3 (one point is inside the triangle and other is not): This case is very similar to case 2, except that there will always be a single point of intersection and PQ cannot be parallel to any of the edge.(Note that the edges and vertices of the triangle are treated as outside region)

1. Determine if PQ intersects any edge of the triangle using Algorithm 2 repeatedly for the 3 edges of the triangle.
2. a) Find the point of intersection using algorithm 3. Let this be X_0.

   b) Theoretically, line segments intersect at a unique point. But because of the pixel based display we see that the line segments actually have more than one point of intersection. To accommodate this and improve the visual appeal we find the portion of line segment PQ around the intersection point which is very close to the corresponding edge of the triangle. This portion of line is determined using Algorithm 4 and coloured as orange to denote the intersection of PQ with the corresponding edge. This portion of line segment is denoted as a line segment defined by 2 points XS_1, XS_2 for X0. Colour the segments XS_1XS_2 as orange and the remainder line segment PQ using green colour.

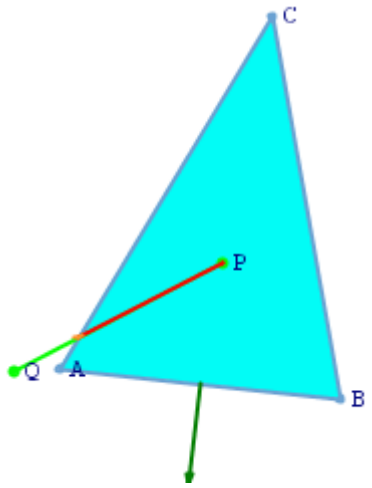

Figure 4

Algorithm 1

Algorithm to determine of a point lies inside, on or outside the triangle

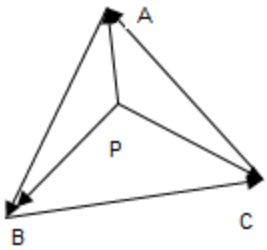

Figure 5

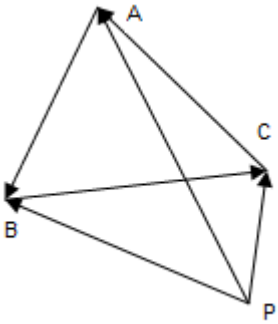

Figure 6

For a point P to lie in triangle ABC, it must satisfy

$$sgn(left(BC).PB) = sgn(left(CA).PC) = sgn(left(AB).PA)$$

else it lies outside the triangle.

Algorithm 2

Algorithm to determine if two line segments intersect each other

Consider 2 line segments AB and PQ. They intersect each other if and only if, for both the line segments, the end points of one line segment are not on the same side of the other line segment.

Mathematically,

$$sgn\left(\overrightarrow{AQ}.left\left(\overrightarrow{AB}\right)\right) \neq sgn(\overrightarrow{AP}.left\left(\overrightarrow{AB}\right))$$

Algorithm 3

Algorithm to determine point of intersection of 2 lines [1]

Let the equation of the lines be

$\overrightarrow{P} = \overrightarrow{P_0} + \mu\left(\overrightarrow{P_0} - \overrightarrow{P_1}\right)$, and

$\overrightarrow{P} = \overrightarrow{Q_0} + \gamma\left(\overrightarrow{Q_0} - \overrightarrow{Q_1}\right)$

Let $\vec{P}$ be the point of intersection of 2 lines and $\vec{N}$ be the normal such that $\overrightarrow{N}.\left(\overrightarrow{Q_0} - \vec{P}\right) = 0$.

Solving the line equations simultaneously and solving for μ, we get

$$\mu = \frac{\vec{N}.(Q_0 - P_0)}{\vec{N}.(P_1 - P_0)}$$

Therefore the point of intersection $\vec{P}$ is given by

$$\overrightarrow{P} = \overrightarrow{P_0} + \frac{\vec{N}.(Q_0 - P_0)}{\vec{N}.(P_1 - P_0)}\left(\overrightarrow{P_0} - \overrightarrow{P_1}\right)$$

Algorithm 4

Algorithm to determine point at a unit distance from a line (AB) on a segment PQ

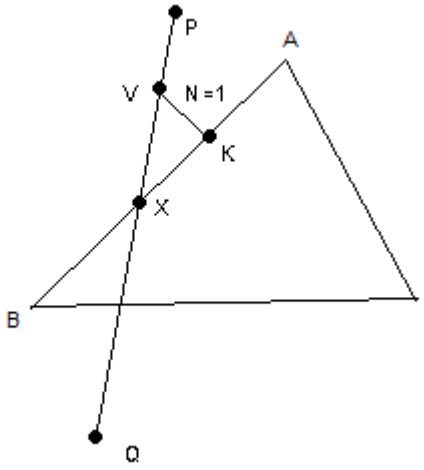

Figure 7

$$\overrightarrow{XV} = \vec{N} + \overrightarrow{XK}$$

$$\Rightarrow \vec{K} = \vec{V} - \vec{N}$$

Also

$$(\vec{K} - \vec{X}).\vec{N} = 0$$

$$\Rightarrow (\vec{V} - \vec{N} - \vec{X}).\vec{N} = 0$$

$$\Rightarrow \vec{V}.\vec{N} = 1 + \vec{N}.\vec{X}\ (as\ \|\vec{N}\| = 1)$$

Let the line segment PQ be represented by the vector line $\vec{X} + \mu(\vec{P} - \vec{X})$

Since $\vec{V}$ lies on the line PQ, $\vec{V} = \overrightarrow{X} + \mu(\vec{P} - \vec{X})$

$$\therefore (\vec{X} + \mu(\vec{P} - \vec{X}).\vec{N} = 1 + \vec{N}.\vec{X}$$

$$\mu = \frac{1}{(\vec{P} - \vec{X}).\vec{N}}$$

$$\vec{V} = \vec{X} + \frac{(\vec{P} - \vec{X})}{(\vec{P} - \vec{X}).\vec{N}}$$

Implementation for extra credit

Algorithm 5

Finding the intersection of 2 triangles

The method described above for finding the intersection between a line segment and triangle can be used to find the intersections between two triangles.

Let the triangles be ABC and PQR. Take each one of the 6 edges one at a time and compute its intersection with the other triangle using the method described above. An example of computed intersection between 2 triangles is shown below:

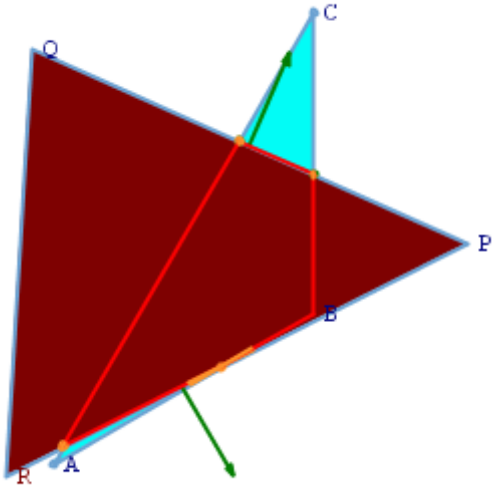

Figure 8

Note: In all the Algorithms which follow, the desired are (intersection, union and difference is shown in GREEN).

Algorithm 6

Finding the intersection of two triangles

Case 1: If one triangle is completely inside the other one, i.e. all three points of a triangle are inside the other triangle then it becomes a trivial case. The intersection is simply the smaller triangle which is inside.

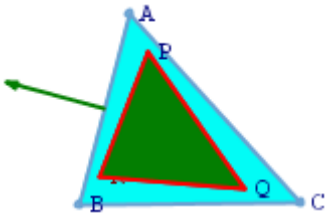

Figure 9

Case 2: We first find the intersection points of each of the three edges (in the order AB, BC, CA) of triangle ABC with triangle PQR. As we calculate the intersections the following logic is used to update the global array (denoted as IntersectionPP[]) which keeps track of all the points of intersection till now:

a) If the edge intersects the other triangle at two points, then we sort the intersection points based on their proximity to the first point of the line segment. Example if we are calculating the intersection of PQ with triangle ABC and the points of intersection are X1 and X2, we find which of the two points X1 and X2 is closer to P and store(append) them in the global array IntersectionPP in the order X1,X2.

b) If the edge intersection the other triangle at a single point X, it means that one point of the edge is inside the other triangle. In this case we add the point inside the triangle to IntersectionPP(global array), followed by the intersection point X. Example if AB intersects PQR at single point X and point B is inside PQR, then we first add point B to IntersectionPP and then X.

Once all three edges have been considered we have the global array IntersectionPP which represents the part of the final required intersection area inside triangle ABC. We now swap the roles of triangle ABC and PQR, i.e. we find the intersection of the edges PQ,QR,RP with triangle ABC in the same fashion and

get another global array IntersectionPP' which represents the part of the final required intersection area inside triangle PQR. If we draw both the global arrays as filled polygons (in GREEN) we get the required intersection area between the two triangles ABC and PQR shaded in GREEN.

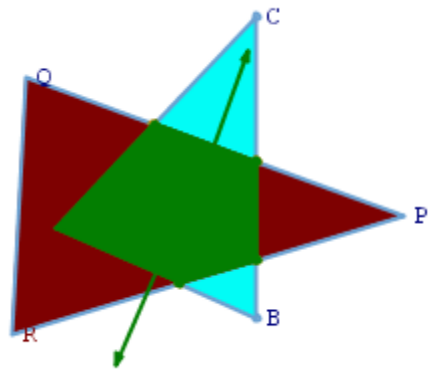

Figure 10

Algorithm 7

Finding the union of two triangles

For finding the union of two triangles, we first calculate the intersection using algorithm 6. Union of triangles ABC and PQR is nothing but both the triangles combined without the intersecting line segments. In algorithm 6 we calculated the intersection points as the arrays IntersectionPP and IntersectionPP'. So to show the union, we simply draw the two triangles first in GREEN colour and then draw the intersection area also in GREEN to delete the intersecting lines segments. So the union is shown in GREEN colour.

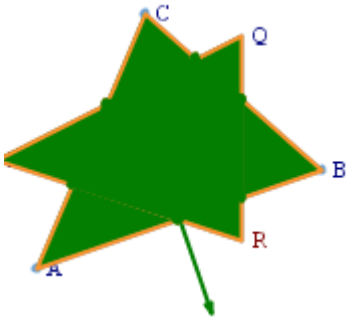

Figure 11

Algorithm 8

Finding the difference of two triangles

Let's say we want to calculate the difference of triangle PQR – triangle ABC: We first calculate the intersection of the two triangles. Now difference is given by triangle PQR without the intersection part. To do this we first show triangle PQR in GREEN and draw the intersection in white (to delete the intersection part). As in previous algorithms, the part painted in green represents the area in interest which is the difference in this case. The screen shot is shown below:

The given program along with this paper calculates triangle PQR – triangle ABC.

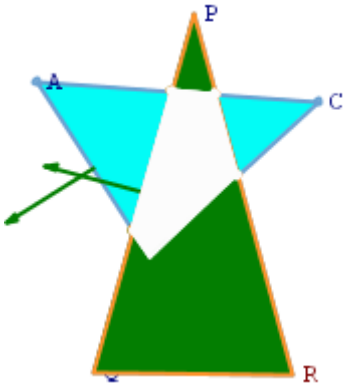

Figure 12

Note for extra credits we have implemented the following:

- Intersection of two triangles
- Unions, intersection and difference of 2 triangles

The Algorithms for all these are given above.

Limitations

1. When 2 line segments overlap, theoretically they intersect at a single point. But on a computer screen we see that line segments have thickness so they intersect at more than one point (contiguous points). We approximate the solution of finding such points by colouring a small segment around the point of intersection. Any such point is within 1 unit (pixel) distance from the intersecting edge of the triangle. If the

triangle and the line segment size becomes very high our approximation will fail and we would see some visual overlap which is not coloured in orange.(Please refer to Algorithm 4 for approximation derivation)

2. Determining the difference of two triangles leaves a white space in the portion where the triangles intersect. It can be corrected by modifying the implementation.

3. In general the algorithm does not take special care about the thickness of lines. To take care of this, we might consider the line as a rectangle and find the intersection of the rectangle with the triangle edge.

References

[1] Dollins, S. C. (2002). Retrieved from Handy Mathematics Facts for Graphics:
http://www.cs.brown.edu/~scd/facts.html

[2] Rossignac, J. Retrieved from CS6491 Foundations of Computer Graphics, Modelling, and Animation:
http://www.gvu.gatech.edu/~jarek/courses/6491/

For Extra Credits we have done the following

- Shown intersection of 2 triangles instead of line segment and triangle
- Calculated unions, intersection and difference areas of 2 triangles