Analysis

This program performs two main queries. The first determines where in a 2D plane two lines intersect and the second tests if the intersection point lines on the desired line. To perform the intersection query parallelism test is required, with collinear lines simply being considered parallel, and then if they are not parallel the intersection point between the two lines is calculated.

**Parallel Query**

The parallel query comes from Farin - Hansford Geometry Toolbox on page 50 eq 3.14. To set up the equation we first find a vector perpendicular to the line1 and then compute the dot product of that vector with the vector from line2. If the dot product is zero then the vectors are perpendicular which means the two input lines are parallel. With this rule collinear lines are simply considered parallel and not treated as a special condition.

This test does require a numerical tolerance. Farin-Hansford recommend using a physically meaningful tolerance so we compute the cos of the angle with the following equation:

C:\Users\lvanhulle\Downloads\CodeCogsEqn (3).gif

Farin-Hansford also recommend a tolerance in the 0.1-0.5 degree range so I chose 0.002 radians or about 0.14 degrees. Any angle less than this will be considered parallel.

**Intersection Point**

After the lines are tested to ensure they are not parallel then the program calculates the intersection point. With the lines in parametric form we know that the intersection point is where the following equation is true:

C:\Users\lvanhulle\Downloads\CodeCogsEqn.gif

With substitution and some algebra we find that the constant t can be solved for giving the equation:

C:\Users\lvanhulle\Downloads\CodeCogsEqn (1).gif

With t known it can be substituted back into the parametric equation which gives us the intersection point.

For the travel path and the pool table sides we want to make sure that this intersection point is on the side of the table (not projected) and in the forward direction of the travel path. If 0 ≤ t ≤ 1 then we know t is between the endpoint of the side. For the travel path we want t > 0. This is the positive direction of the travel path. The ‘is point on line’ property is not tested for the table sides since, due to the setup of the problem, an intersection never occurs within 1 unit of an end unless it is the very last collisions (the ball goes into the pocket).

**Is point on line segment?**

Since the line segment PQ does not cause integer collisions and we decided that end points should not count as crossings it requires some additional testing to ensure that a calculated intersection is actually on itself. For the tolerance of this equation I decided to use an epsilon that was relative to the length of the table’s diagonal. This was done so that as the table got larger imprecision of t’s calculation would not cause unreliable results.