Hana Park CS 470 Exam 4 Analysis

Question 1: Given two set of line segments, {[P11, P12, P13, P14, …,P1(N-1), P1(N)}, and {[P21, P22, P23, P24, …,P2(N-1), P2(N)}. They represent two ***convex*** polygons, and the vertices are in clockwise order. Please check if these two polygons intersect each other.

1. **Implementation Explanation:**

In order to determine if two polygons intersect, I compared every segment created by the first polygon and saw if it intersected with every segment in the second polygon. If two segments intersect, that means the direction of each point in the first segment is opposite of the other point in relation to the segment created by the second polygon line segment. The Dir function helps determine this by finding the cross product of the second polygon line and the line created by the point in the second polygon and the actual point being compared. This result shows the orientation of the point in relation to the segment. If the orientation of the two points are opposite (1 being negative and 1 being poisitive) that means that they intersect. This methodology is seen in function SegInt.

By using the methodology above for every line combination from polygon1 and polygon 2, the intersection status of the polygons can be determined by if any of the line combinations intersect. This methodology works on both concave and convex polygons, but "CS470E4Q1Convex.cpp" below only works on convex polygons by seeing if any point in polygon 1 resides in polygon 2 and vice-versa.

2. **Time and Space Complexity Analysis**

To determine the time complexity of the program, we will add all the running times of each function. For N points in polygon 1 and M points in polygon 2, each comparison will take exactly O(1) time because the determination of line intersection is just simple calculation. Because it take M*N comparisons, the final time complexity is O(M*N).

Because all data values are stored in an array list relative to the number of points in the polygons. The space complexity is O(M+N).

3. **The screenshots of your algorithms' outputs**

Input	Output

`cout<<"Test Case 1"<<endl;`
`Point polygon1[] = {{61,100},{99,84},{76,78},{62,64}};`
`Point polygon2[]={{8,58},{74,55},{71,27},{64,11},{4,21}};`
	`Test Case 1`
	`Polygons do not intersect`

`cout<<"Test Case 2"<<endl;`
`Point polygon1[] = {{59,94},{78,100},{100,59},{58,68},{51,73}};`
`Point polygon2[]={{31,48},{69,39},{52,10},{16,19},{13,35}};`
	`Test Case 2`
	`Polygons do not intersect`

`cout<<"Test Case 3"<<endl;`
`Point polygon1[] = {{31,63},{74,96},{62,33},{27,20},{29,54}};`
`Point polygon2[]={{32,63},{73,70},{67,55},{54,37},{30,9},{2,19}};`
	`Test Case 3`
	`Polygons Intersect`

CS470E4Q1Convex.cpp

```cpp
//Hana Park
//Question 1
//Only works with convex polygons by seeing if points are in each
#include<iostream>
#include <climits>
using namespace std;

struct Point {
   int x, y;
};

struct line {
   Point p1, p2;
};

bool onSegment(line l1, Point p) {        //check whether p is on the line or not
   if(p.x <= max(l1.p1.x, l1.p2.x) && p.x <= min(l1.p1.x, l1.p2.x) &&
      (p.y <= max(l1.p1.y, l1.p2.y) && p.y <= min(l1.p1.y, l1.p2.y)))
         return true;

   return false;
}

int Dir(Point a, Point b, Point c) {
   int val = (b.y-a.y)*(c.x-b.x)-(b.x-a.x)*(c.y-b.y);
   if (val == 0)
      return 0;           //colinear
   else if(val < 0)
      return -1;          //counter-clockwise direction
      return 1;          //clockwise direction
}

bool isIntersect(line l1, line l2) {
   //four direction for two lines and points of other line
    int d1 = Dir(l1.p1, l1.p2, l2.p1); //line 2 point 1 on line1
   int d2 = Dir(l1.p1, l1.p2, l2.p2); //line 2 point 2 on line1
   int d3 = Dir(l2.p1, l2.p2, l1.p1);  //line 1 point 1 on line2
   int d4 = Dir(l2.p1, l2.p2, l1.p2); //line 1 point 1 on line2

   if(d1 != d2 && d3 != d4)
      return true;           //they are intersecting
   if(d1==0 && onSegment(l1, l2.p1))         //when p2 of line2 are on the line1
      return true;
   if(d2==0 && onSegment(l1, l2.p2))         //when p1 of line2 are on the line1
      return true;
   if(d3==0 && onSegment(l2, l1.p1))          //when p2 of line1 are on the line2
      return true;
   if(d4==0 && onSegment(l2, l1.p2))         //when p1 of line1 are on the line2
```

```
      return true;
   return false;
}

bool inside(Point poly[], int n, Point p) {
   if(n < 3)
      return false;                  //when polygon has less than 3 edge, it is not polygon
   line exline = {p, {20000, p.y}};   //create a point at infinity, y is same as point p
   int count = 0;
   int i = 0;
   do {
      //side 1: point1 to point2, point 2 to point 3, etc...
      line side = {poly[i], poly[(i+1)%n]};     //forming a line from two consecutive points
of poly
      if(isIntersect(side, exline)) {          //if side intersects exline
         if(Dir(side.p1, p, side.p2) == 0)      ////if colinear, check if p on side line of
polygon
            return onSegment(side, p);
         count++;
      }
      i = (i+1)%n;
   } while(i != 0);
      return count&1;             //when count is odd
      //when count is odd (last bit ends with 1) & with 1 to get 1
      //count being odd indicates that the count number of sides of polygon are on
      //the right side of the point (on horizontal line)
      //odd number indicates that there is a side on the left side of the point,
      //thus encapsulating point
}


int main() {
   // line polygon = {{{0,0},{10,0}},{{10,0},{10,10}},{{10,10},{0,10}},{{0,10},{0,0}}};
//    Point polygon1[] = {{0, 0}, {10, 3}, {12, 10}, {0, 10}};
//    Point polygon2[] = {{5, 0}, {15, 3}, {17, 10}, {5, 10}};
   // Point polygon1[] = {{0, 0}, {10, 3}, {12, 10}, {0, 10}};
   // Point polygon2[] = {{5, 11}, {15, 14}, {17, 21}, {5, 21}};
   //  cout<<"Test Case 1"<<endl;
   //  Point polygon1[] = {{61,100},{99,84},{76,78},{62,64}};
   //  Point polygon2[]={{8,58},{74,55},{71,27},{64,11},{4,21}};
   //  cout<<"Test Case 2"<<endl;
   //  Point polygon1[] = {{59,94},{78,100},{100,59},{58,68},{51,73}};
   //  Point polygon2[]={{31,48},{69,39},{52,10},{16,19},{13,35}};
    cout<<"Test Case 3"<<endl;
    Point polygon1[] =  {{31,63},{74,96},{62,33},{27,20},{29,54}};
    Point polygon2[]={{32,63},{73,70},{67,55},{54,37},{30,9},{2,19}};

   Point p = {0, 0};
   int num_vertexes_p1 = 4;
   int num_vertexes_p2 = 4;
```

```
   bool P1pointsinP2=false;
   bool P2pointsinP1=false;
   for(int i=0;i<num_vertexes_p1;i++){
        if(inside(polygon2, num_vertexes_p2, polygon1[i])){
             //cout << "Point of Polygon1 in Polygon2"<<endl;
             P1pointsinP2=true;
             break;
        }
   }
   for(int i=0;i<num_vertexes_p2;i++){
        if(inside(polygon1, num_vertexes_p1, polygon2[i])){
             //cout << "Point of Polygon2 in Polygon1"<<endl;
             P2pointsinP1=true;
             break;
        }

   }

    if(P1pointsinP2||P2pointsinP1){
      cout << "The Polygons Intersect";
      }
   else{

      cout << "The Polygons Do Not Intersect";
   }
}
```